VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Informační systém pro zpracování a schvalování žádostí o přidělení výpočetního času a správu uživatelů přistupujících k prostředkům superpočítačového centra

Information System for User Self Management and Proposal Processing at Supercomputing Center

2015 **Bc. Lumír Balhar**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Lumír Balhar**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Informační systém pro zpracování a schvalování žádostí o přidělení výpočetního času a správu uživatelů přistupujících k prostředkům superpočítačového centra.
Information System for User Self Management and Proposal Processing at Supercomputing Center

Zásady pro vypracování:

Cílem diplomové práce je rozsáhlý informační systém zakomponovaný do současného ekosystému nástrojů využívaných v rámci superpočítačového centra IT4Innovations.
Informační systém nabídne uživatelům následující možnosti v závislosti na jejich rolích.
1. Pro přístup k systému budou využity existující účty uživatelů. V případě nového uživatele bude umožněna registrace nového účtu splňující adekvátní bezpečností kritéria včetně ověření totožnosti registrovaného uživatele.
2. Žadatelům o schválení přístupu jejich projektu k výpočetním prostředkům superpočítačového centra bude umožněno zpracovat online žádost a její následná editace až do finálního odeslání schvalovací komisi.
3. Členům schvalovací komise bude umožněno si zobrazit, komentovat a hodnotit jednotlivé žádosti.
4. Uživatelům superpočítačového centra bude umožněno spravovat svůj účet a sledovat podrobnosti o projektech, k nimž jsou přiřazeni.
5. Novým uživatelům superpočítačového centra bude umožněna registrace společně s podáním žádosti o přiřazení k existujícímu projektu.
6. Hlavním řešitelům projektu bude umožněno spravovat svůj vlastní účet, přiřazené projekty, schvalovat žádosti uživatelů o přiřazení k jejich projektu a přiřazovat k projektu již existující uživatele.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr. Branislav Jansík, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

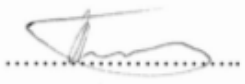
doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 5. 5. 2015



Podpis

Poděkování

Zde bych rád vyjádřil poděkování těm, kteří přispěli ke zdárnému dokončení této diplomové práce a jmenovitě pak panu Mgr. Branislavu Jansíkovi, Ph.D. za vedení této práce.

Abstrakt

Tato diplomová práce se pokouší navázat na zaběhnutý provoz největšího superpočítačového centra v České republice a s pomocí znalostí jeho stávajících procesů, vnitřního fungování, znalostí z oblasti vývoje softwarových aplikací, bezpečnosti a plánování procesů navrhnout a implementovat informační systém podporující provoz superpočítačového centra a usnadňující provádění opakovaných činností jeho řídících struktur.

Práce je koncipována jako kompletní návrh s popisem realizace nového informačního systému, použitých softwarových komponent a nástrojů pro jeho vývoj a správu s důrazem na teoretický rozbor jednotlivých částí.

Klíčová slova

Superpočítačové centrum, informační systémy, bezpečnost, procesy, software

Abstract

This thesis is trying to extend a running operation of the largest supercomputing centre in the Czech Republic and with the knowledge of its existing processes, internal functioning, software application development, security and planning processes to design and implement an information system supporting the operation of the supercomputer centre and facilitating the implementation of recurring activities of its managing structures.

The work is conceived as a complete proposal describing the implementation of a new information system, used software components and tools for its development and management with an emphasis on the theoretical analysis of the individual parts.

Keywords

Supercomputing centre, information systems, security, processes, software

Seznam použitých zkratek

- SCC - Supercomputing center
- CRL - Certificate Revocation List
- MVC - Model View Controller
- LDAP - Lightweight Directory Access Protocol
- SSH - Secure Shell
- SCCS - Supercomputing center system
- API - Application Programming Interface
- HTTP - Hypertext Transfer Protocol
- SSL - Secure Sockets Layer
- HTTPS - Hypertext Transfer Protocol Secure
- IS - Information system
- SQL - Structured Query Language
- MIMT - Man in the middle
- XSS - Cross-site scripting
- XML - Extensible Markup Language
- TAL - Template Attribute Language

Obsah

1. Úvod

Cílem této práce je navázat na existující systém procesů v superpočítačovém centru, podrobit je analýze a pokusit se o jejich zefektivnění s pomocí vývoje nového informačního systému, který umožní jednodušeji provádět rutinní úkoly pracovníků centra a také předá část pravomocí do rukou uživatelům, kteří tak budou moci obstarávat část vlastní agendy samostatně.

Práce se snaží stavět na znalosti současného fungování centra a úzkou spolupráci s jeho vedením.

Nedílnou součástí je teoretická část, ve které je problematika vývoje software a použití jednotlivých součástí rozebrána teoreticky a na kterou je následně navázáno při specifikaci, implementaci a výběru softwarových komponent a podpůrných systémů.

2. Motivace k tvorbě IS

Při tvorbě jakéhokoli informačního systému je nejdůležitějším prvním krokem zamyšlení se nad motivací k samotné tvorbě a cílem, kterého se snažíme tvorbou dosáhnout. Čím důkladněji si promyslíme jednotlivé kroky vývoje a očekávané výsledky, tím větší je pravděpodobnost, že bude výsledný produkt užitečný a práce na něm nebude ztrátou času.

V tomto konkrétním případě je hlavní motivací nutnost restrukturalizace procesů, které jsou v rámci superpočítačového centra používány především v oblasti komunikace s uživateli superpočítače, potřeba osamostatnění uživatelů v některých rutinních činnostech a zlepšení a zrychlení některých interních procesů SCC.

2.1 Historie tvorby pracovních postupů

Superpočítač Anselm byl spuštěn v květnu 2013 a krátce po jeho uvedení do provozu se týmu správců začaly kupit požadavky vycházející z potřeb uživatelů. Většina z nich byla rutinního charakteru a bylo třeba je rychle vyřešit a ruku v ruce s tím vymyslet pracovní postupy, které by tyto opakující se procesy podchytily v samotném počátku.

Problémem ovšem byl nedostatek času, který značně omezoval možnosti vývoje různých podpůrných nástrojů a nutil tak pracovníky k využití existujících a ne vždy zcela vyhovujících nástrojů, které se v případě potřeby danému účelu přizpůsobily. Takové změny byly o to složitější, že bylo třeba dodržet celou řadu mechanismů pro zachování bezpečnosti a důvěryhodnosti sytému jako celku.

V následujících kapitolách si popíšeme současně nejpoužívanější pracovní postupy, jejich kladné a záporné stránky a možnosti, jejich optimalizace a částečné automatizace. V dalších částech této práce bude následovat popis implementace informačního systému, který by měl vyřešit řadu problematických pracovních procesů.

2.2 Současné pracovní postupy

V následujících podkapitolách se pokusím stručně popsat současné aktuální pracovní postupy tak, jak z historického hlediska vznikly a jak se aktuálně používají.

Vzhledem ke snaze o jejich nahrazení nebudou tyto postupy doplněny o jejich grafické znázornění, které pro dokumentační účely doplním až v následujících kapitolách, jež se zabývají zavedením nových postupů a jejich implementací do informačních systému superpočítačového centra.

2.2.1 Podávání projektových žádostí

Podávání projektových žádostí je jedním z hlavních procesů celého superpočítačového centra. Do těchto žádostí budoucí řešitelé projektů zahrnou všechny hlavní aspekty jimi řešeného problému, množství a typ požadovaných zdrojů a očekávané výsledky výzkumu.

Formulář pro projektovou žádost se s každým kolem může měnit a jeho aktuální verze je na začátku dané výzvy vždy vystavena ke stažení na webových stránkách organizace.

Samotný dokument ještě pro uživatele neznamená žádné velké riziko a naopak se snaží přesně

vytyčit informace, které bude alokační komise potřebovat ke kvalifikovanému rozhodnutí.

Problematický je ovšem současný způsob doručení formuláře, kdy žadatelé odesílají vyplněnou žádost mailem řediteli SCS. Mail je s ohledem na své principy fungování nevhodný pro přenos takto velkých souborů s citlivým obsahem a jeho správné doručení není vždy garantováno. Vezmeme-li v úvahu i velikost agendy ředitele SCS, nelze od něj požadovat potvrzování přijetí každé jednotlivé žádosti, ať už jakoukoli formou.

Další základní překážkou jsou možné následné změny jednotlivých žádostí, které vyžadují jejich opětovné odeslání i s informací, zda se jedná o přepracovanou žádost nebo jen o nějakou marginální změnu.

Tento způsob je tedy nutné převést do informačního systému, snížit zátěž na zpracování a lidské zdroje a také přinést uživatelům vyšší komfort při podávání žádostí.

2.2.2 Hodnotící komise

Po doručení všech projektových žádostí a po termínu pro jejich podávání přichází na řadu jejich hodnocení hodnotící komisí.

Tvorba hodnotící komise aktuálně probíhá v tabulkovém editoru, do kterého jsou zaneseny hlavní informace o jednotlivých žádostech. Tvůrce alokační komise, kterým je zpravidla ředitel SCS, následně přiřadí ke každé komisi dva hodnotitele a do tabulky doplní na dané místo jejich identifikátory.

Takto připravený dokument následně odešle všem členům hodnotící komise, kteří jednotlivé žádosti okomentují a také jim přiřadí číselné hodnocení.

Komentování a hodnocení probíhá ve třech předem daných oblastech:

- připravenost z vědeckého pohledu,
- připravenost pro výpočet,
- socioekonomický dopad projektu.

Po ukončení hodnocení jsou jeho výsledky předány zpět řediteli SCS, který je pak dále zpracovává.

2.2.3 Alokační komise

Jakmile se dokument popsaný v předchozí kapitole vrátí s očekávanými výsledky zpět k řediteli SCS, ten jeho obsah spolu s dalšími členy alokační komise vyhodnotí. Pro jednoduchost obsahuje dokument zmíněné číselné hodnocení, nicméně v případě nejasností mají členové alokační komise při finálním rozhodnutí možnost zjistit příčinu konečného hodnocení právě v komentáři k dané oblasti.

Na základě hodnocení hodnotících komisí pro jednotlivé žádosti a vlastního úsudku se pak alokační komise rozhodne, zda projekt schválí či nikoli. Často se také stává, že dojde ke schválení projektu, ale také k úpravě požadované alokace, pokud pro její rozsah nebyl shledán dostatečný důvod.

O rozhodnutí alokační komise jsou následně informování žadatelé. Tento zpětný informační email také obsahuje instrukce, jak dále postupovat při získávání přístupu na Anselm, ať již pro hlavního řešitele, nebo pro jeho spolupracovníky.

Projekt je následně zaveden do existujícího systému pro evidenci projektů se všemi důležitými parametry. Mezi ty hlavní patři již zmíněná alokace, druh přidělených zdrojů, začátek a konec čerpání

alokace, jednoznačný identifikátor projektu atp.

Takto je projekt připraven pro přiřazení jednotlivých uživatelů.

Motivace ke zlepšení a automatizaci procesů je v tomto případě zcela zřejmá. Fungování alokační komise pomocí sdíleného dokumentu je možná jednoduché pro vytvoření, ale ve finále to přináší velkou zátěž na lidské zdroje a problémy spojené s volbou tohoto prostředku komunikace.

2.2.4 Schvalování uživatelů k projektu

Po schválení projektu a jeho zavedení do existujícího informačního systému přijde na řadu další fáze, kdy hlavní řešitel projektu schválí jednotlivé uživatele ke svému projektu.

Je na uvážení hlavního řešitele, zda se mezi ně zařadí či nikoli.

Tato schvalovací žádost je odeslána do ticketovacího systému, ve kterém je převzata některým z pracovníků helpdesku a zaevidována.

Pokud se v žádosti objeví uživatelé, kteří již na superpočítači konto mají, jsou ihned přiřazeni k novému projektu. V opačném případě slouží tento ticket pro evidenci a čeká do chvíle, kdy si uživatel o konto požádá.

Vzhledem k tomu, že se zde již finálně schvalují jednotlivá konta uživatelů, je nezbytné dbát na bezpečnost a především na ověření identity hlavního řešitele projektu. Proto je nutné žádost respektive email podepsat platným certifikátem vydaným uznávanou certifikační autoritou. Tento elektronický podpis je následně po přijetí v systému verifikován a to nejen na platnost a důvěryhodnost, ale také proti aktuálním CRL seznamům dané certifikační autority.

Pro většinu zkušenějších uživatelů není získání a instalace certifikátu problém, často se ovšem objevují uživatelé, kteří si s tímto požadavkem nevědí rady a potřebují buď detailně poradit, nebo provést verifikaci své žádosti nějakou alternativní cestou, kterou může být například osobní návštěva.

I přes to, že je tento způsob ověření žádosti pro většinu hlavních řešitelů schůdný, není tomu tak vždy a žádost jako taková je často jediným důvodem, proč si hlavní řešitelé a následně uživatelé certifikát vyřizují.

Další zápornou vlastností je zátěž na pracovníky podpory, kteří žádosti jako takové zpracovávají a přiřazují uživatele k projektům.

2.2.5 Tvorba uživatelských kont

Založení nového uživatelského konta na superpočítači probíhá velmi podobně jako schvalování uživatelů k projektu.

Na začátku celého procesu je opět žádost zaslaná do *ticketovacího* systému, kde ji musí některý z pracovníků podpory převzít a vyřídit.

Žádost musí mimo jiné obsahovat jisté náležitosti, mezi které se řadí především informace o uživateli a instituci, ze které pochází, identifikátor projektu, ke kterému se hlásí, a jeho preferované uživatelské jméno.

Nedílnou součástí žádosti je také přiložený dokument s názvem *Acceptable use policy* (dále AUP), který je k dispozici ke stažení na webových stránkách a se kterým musí uživatel vyslovit souhlas.

Dokument obsahuje podmínky používání superpočítačových služeb a pravidla chování při využívání superpočítače.

Stejně jako u žádosti o přiřazení uživatele k projektu je i zde nezbytnou součástí certifikát, pomocí něhož uživatel žádost podepíše. Kromě ověření identity uživatele se zde certifikát ovšem používá i pro šifrování odpovědi uživateli, která obsahuje přihlašovací údaje a klíče pro přístup na superpočítač.

Vzhledem k širšímu využití certifikátu se v tomto procesu dost obtížně hledají alternativní cesty doručení uživatelských údajů uživatelům, kteří si nedokázali certifikát zřídit.

Pokud tělo žádosti obsahuje všechny nezbytné položky a certifikát je dle systému platný, může pracovník podpory přistoupit k vytvoření uživatelského konta.

V prvním kroku dojde ještě k systémovému ověření pravosti výše zmíněného dokumentu AUP, aby se prověřilo, zda od jeho získání z webových stránek nedošlo ke změně jeho obsahu nebo nebyla použita starší verze dokumentu. K tomuto účelu se využívá kontrolní součet MD5.

Následně pracovník podpory spustí skript, kterému předá potřebné informace formou parametrů a tento skript vytvoří uživatelský účet se všemi náležitostmi a odešle informace o něm šifrovanou zprávou zpět uživateli.

Kromě opět se opakujících potíží s certifikáty je zde velký prostor pro optimalizaci a automatizaci celého procesu, kterých se pokusím v informačním systému dosáhnout.

2.3 Návrh na zlepšení pracovních postupů

Jednou z hlavních úloh nově vznikajícího informačního systému je odhlehčení rutinních úkolů pracovníkům podpory superpočítačového centra a přesun agendy na samotné uživatele.

V kapitole zabývající se specifikací nového informačního systému se tedy budeme snažit přesně definovat chování a používání informačního systému tak, aby v maximální možné míře pokryl a redefinoval historické a dnes již nevyhovující procesy v superpočítačovém centru, přenesl důležitou agendu na uživatele a administrátorům i vedení tohoto centra v maximální možné míře usnadnil práci a zlepšil jejich komfort.

3. Teoretická část návrhu IS

V následujících kapitolách se budeme zabývat teoretickým pohledem na návrh informačního systému, metody jeho vývoje a výběr jeho nejzákladnějších komponent.

U každé podkapitoly se přitom pokusíme nepřihlížet ke svým dosavadním zkušenostem a oblíbeným postupům či nástrojům, ale budeme se snažit zaměřit se na danou problematiku laickým a co možná nejširším pohledem, abychom si mohli být jisti, že rozhodnutí budou opravdu podložená aktuálními potřebami a fakty a nikoli předchozími zkušenostmi, které by rozhodování mohly ovlivnit.

3.1 Postup vývoje IS

Metodiky vývoje software v sobě obsahují celou řadu postupů, nástrojů a pravidel, které se při konkrétním vývoji používají napříč celým týmem pracovníků či celou organizací pro návrh, plánování a řízení vývoje daného software či v našem případě informačního systému.

V současné době existují dva hlavní směry metodik vývoje software - jedná se o rigorózní a agilní metodiky.

Rigorózní metodiky jsou konkrétnější a obsáhlejší ve fázi návrhu a popisují jednotlivé procesy vývoje, definují posloupnost činností a jsou založeny na sériovém (vodopádovém) přístupu k vývoji. Některé rigorózní metodiky jsou realizovány iterativním přístupem, což znamená, že se v nich některé fáze vývoje několikrát opakují. Zpětná vazba se v rigorózních metodikách mezi jednotlivými fázemi přenáší pomocí systémů řízení změn.

Mezi nejznámější zástupce rigorózních metodik vývoje softwaru se řadí následující:

- Vodopádový model - jedná se o jednu z nejstarších metodik, která se dnes již příliš nepoužívá. Tato metodika využívá několik fází - specifikace, návrh, implementace, integrace, údržba - mezi kterými nelze přeskakovat, ani se vracet do předchozí fáze. Jediným povoleným skokem je možnost přejít z údržby zpět k návrhu. Jedná se o nepružnou metodiku, která nemusí přesně reflektovat požadavky uživatele, místo toho generuje velké množství dokumentace.
- Spirálový model - jde o komplexní metodiku vhodnou pro velké projekty. Metodika obsahuje několik fází - plánování, stanovení cílů, analýza rizik, vývoj a testování - a tyto fáze se opakují pro jednotlivé části vývoje software, jako jsou: vytváření konceptu, tvorba specifikace, tvorba návrhu a architektury, tvorba designu, implementace, testování apod. Před každým samotným vývojem dojde v této metodice k vytvoření prototypu a velmi často zde dochází ke zvažování alternativ.
- *Rational Unified Process* (RUP) - jedná se o velmi rozsáhlou a placenou metodiku vývoje software, která obsahuje mimo jiné i nástroje k vytvoření metodiky na míru. Celá metodika je objektově orientovaná a jako svůj základ používá případy užití. Z tohoto pohledu jsou v metodice jako základní stavební kameny použity hlavní objekty reprezentace osob a činností. Metodika jako celek obsahuje celkem čtyři fáze: zahájení (10 %), projektování (30 %), realizace (50 %), předání (10 %).

Tradiční rigorózní přístup k vývoji software se jak vidno zaměřuje na hlubokou analýzu, přesný

popis a dokonalý návrh řešení pro daný projekt, ale v posledních letech je vývoj software velmi dynamicky se měnící disciplína, což vyžaduje i změny na poli metodik.

V současné době se tak začaly objevovat metodiky vývoje software, které na celý proces vývoje pohlížejí flexibilně a jsou schopny reagovat na měnící a vyvíjející se požadavky na software i v průběhu jeho vývoje. Agilní metodiky umožňují vytvářet softwarové produkty velmi rychle a pružně je přizpůsobovat měnícím se požadavkům. Kromě rychlého vývoje je dalším pilířem agilních metodik zpětná vazba přicházející přímo od koncového uživatele.

Mezi nejznámější agilní metodiky vývoje se řadí následující tři:

- Extrémní programování - jedná se o pravděpodobně nejznámější agilní metodiku vývoje, která prosazuje několik základních pilířů: časté dodávky software v krátkých vývojových cyklech, párové programování, *test driven development* a programování pouze toho, co je v danou chvíli nezbytné. Důležitá je zde častá komunikace mezi programátorem a zákazníkem a také mezi samotnými programátory.
- Scrum - klíčovou částí metodiky jsou setkání programátorů na denní bázi nazývaná scrum. Každý člen týmu na této schůzce referuje o své činnosti z minulého dne, o plánech na aktuální den a o problémech, na které při vývoji narazil. V metodice se používá vývoj pomocí tzv. sprintů, kdy každý sprint trvá 2 až 4 týdny a jehož výsledkem je demo, které je předvedeno zákazníkovi. Ten pak díky takovému demu poskytne vývojářům zpětnou vazbu a ti na ni mohou rychle reagovat.
- *Feature Driven Development* (FDD) - vývoj řízený vlastnostmi - FDD začíná vytvořením doménového modelu, který popisuje základní systém a z něj vycházející základní funkcionality pro uživatele. Po rozdělení modelu na části a jejich přidělení programátorům probíhá vývoj iterativně ve zpravidla dvoutýdenních cyklech, kdy jsou během cyklu navrhovány a implementovány užitné vlastnosti systému a na konci každého cyklu jsou výsledky prezentovány zákazníkovi.

S ohledem na dynamicky se měnící procesy v organizaci a jejich potřebu rychlé implementace do nově vznikajícího informačního systému jsme se rozhodli přiklonit k některé z agilních metodik vývoje. Při výběru některé z nich jsme zvážili především charakter a frekvenci přicházejících požadavků a jejich zaměření na nově vznikající funkcionality v daných fázích vyhlášených výzev pro podávání žádostí a rozhodli jsme se pro použití metodiky Vývoje řízeného vlastnostmi (FDD). Hlavní části této metodiky jsme pak uvedli do praxe a s menšími obměnami se jimi řídil celý vývoj nově vznikajícího informačního systému.

3.2 Výběr databázové vrstvy

Databáze je datová základna softwarového produktu, která obsahuje strukturovanou množinu dat, která jsou uložena na nějakém paměťovém médiu a jsou prostřednictvím podpůrných nástrojů dále poskytována dalším částem aplikace.

Nejjednodušším databázovým systémem, který se při určitém úhlu pohledu obejde i bez systému řízení báze dat, může být například ukládání dat do textových či binárních souborů na souborovém systému. V dnešní době máme ovšem k dispozici daleko sofistikovanější a výkonnější řešení.

Mezi nejznámější databázové modely patří tyto tři:

- Hierarchická databáze využívá pro ukládání dat hierarchickou stromovou strukturu a byla prvním databázovým typem hojně rozšířeným do praxe. Postupem času se však objevily nedostatky při modelování reality - především nemožnost jednoduché tvorby vazeb M:N - což vedlo k opuštění tohoto konceptu, který byl nejdříve nahrazen síťovou a posléze relační databází.
- Relační databáze jsou asi nejznámějším zástupcem na poli databázových modelů. Jsou založeny na relačním modelu a data jsou ukládána do tabulek, které jsou pomocí primárních a cizích klíčů mezi sebou propojeny do relací. Tyto tabulky mají jednoznačně definované schéma a pro získávání dat a jejich modifikaci se využívá jazyk SQL.
- Objektová databáze se vyvíjí ruku v ruce s přechodem k objektově orientovanému programování, při kterém vyvstala potřeba ukládat objekty jako celky, aby se s nimi dobře pracovalo v aplikační části a nemusela se příliš řešit vazba mezi objektem v aplikaci a jeho reprezentací v databázi. Jedná se často o velmi optimalizované databáze typu klíč - hodnota.

Vzhledem k charakteru dat, selektivitě jejich výběru a nutnosti použití mnoha různých pohledů a relací mezi jednotlivými daty jsme se rozhodli použít relační databáze, která svým principem dobře pokryje všechny nároky aplikace na ukládaná data a práci s nimi.

Při pozdějším výběru konkrétního řešení pro databázovou vrstvu bude muset být s ohledem na výkonnost a dostupnost brán zřetel také na možnosti škálování, stabilitu a výkonnost takového řešení.

3.3 Výběr webového serveru

Webový server je označení pro fyzický počítač zastávající tuto funkci, nebo pro jeho softwarové vybavení. My se budeme samozřejmě zabývat druhou možností a tedy základní charakteristikou webového serveru jako démona, soupisem nejpoužívanějších softwarových řešení a následně výběrem toho nejvhodnějšího pro nasazení do produkčního prostředí.

Webový server jako takový je aplikace, která umožňuje zachytávat, zpracovávat a reagovat na požadavky přicházející z webových prohlížečů (klientů).

Webový server přijímá od klientů požadavky ve tvaru specifikovaném protokolem HTTP a následně na ně odpovídá jednak stavem popisujícím výsledek zpracovaného dotazu a pak také samotným poskytnutým obsahem, kterým může být obsah HTML dokumentu, obrázek či jakýkoli jiný typ souboru.

Obsah poskytnutý webovým serverem na základě požadavku klienta může být buď statický, a tedy přečten z nějakého paměťového média a bez změny odeslán klientovi, nebo dynamicky generován nějakým programem, který může zpracovat různé části uživatelského vstupu a na jejich základě vygenerovat dynamický výstup, který se následně odešle uživateli. Dynamické generování je už z principu výpočetně náročnější a pomalejší, ale umožňuje generování daleko většího množství obsahu, než kdyby ten byl celý předem vygenerován staticky. Tyto dvě zmíněné metody lze i kombinovat tzv. *kešováním*, kdy je dynamicky vygenerovaný obsah po nějakou dobu uložen ve statické podobě a na stejný příchozí dotaz neprobíhá jeho nové generování, čímž se ušetří čas a výpočetní prostředky.

Kromě součástí, které mají webservery společné, jako jsou například chybové a potvrzovací odpovědi na požadavky, se webové servery v mnoha ohledech liší. Největší rozdíly lze najít zejména mezi množstvím poskytovaných funkcí nad rámec běžného přenosu dat protokolem HTTP, jako může být například různé přesměrování mezi URL adresami, schopnost zpracování a vynucení SSL šifrování

a použití SSL certifikátů, použití různých metod generování dynamického obsahu, kešování, balancování provozu mezi několika zdroji, komprese dat, streamování videa, nastavitelné logování apod.

Mezi nejznámější a na internetu nejrozšířenější webové servery patří tyto:

- Apache - Jedná se o nejpoužívanější webový server vůbec. Škála jeho možností je velmi široká a je také snadno rozšiřitelná pomocí dodatečných modulů.
- IIS je druhý nejpoužívanější webový server na internetu, který je vyvíjen společností Microsoft a je tedy dostupný pouze pro operační systémy od této společnosti.
- Nginx je webový server pro unixové systémy, který pracuje asynchronně a pro zpracování požadavků využívá jen jedno vlákno. Neimplementuje tolik funkcionality jako první zmíněný webserver, ale snaží se o maximální rychlost a nejmenší možnou náročnost na systémové prostředky. Je velmi škálovatelný a často se používá pro distribuci webového provozu mezi dalšími systémy v roli proxy serverů.
- Lighttpd je také jeden z minimalistických webových serverů, který stejně jako Nginx pracuje asynchronně a jeden hlavní proces zpracovává požadavky pomocí vzniklých vláken pro každý z nich. Lighttpd implementuje přibližně stejnou množinu funkcionalit jako Nginx, ale bohužel není tak rychlý. Z tohoto důvodu prozatím nedošlo k jeho masivnímu rozšíření.

Při zvážení všech výhod a nevýhod poskytovaných řešení a také s ohledem na neustále se měnící požadavky, agilní způsob vývoje a nutnost pokrytí širokého spektra funkcionalit rozhodli jsme se pro webový server Apache, který sice není ve zmíněné množině nejrychlejší, ale poskytuje zdaleka nejširší množství funkcí.

3.4 Skriptovací jazyky vhodné pro vývoj IS

Skriptovací jazyky jsou interpretované programovací jazyky, u nichž se klade důraz na jednoduchost a rychlost vývoje. Neexistuje žádná přesná definice skriptovacích jazyků, ale přesto mají několik společných vlastností, které je od ostatních odlišují a jsou pro ně typické. Mezi tyto vlastnosti patří:

- Skriptovací jazyk je interpretovaný, což znamená, že k běhu programu v něm napsaném je třeba zdrojový kód a interpret, který jej dokáže spustit, resp. interpretovat. Protipólem jsou kompilované jazyky, které se nejdříve překladačem zkompilují do strojového kódu, který je pak již možné spustit bez potřeby vlastnit překladač nebo interpret.
- Není zde třeba deklarovat proměnné, což značně zjednodušuje práci programátora.
- Je v nich možné automaticky konvertovat data mezi jednotlivými datovými typy díky dynamické typové kontrole.
- Skripty jsou schopné se zotavit z chyb, které neústí v jejich ukončení.
- U práce se složitějšími datovými typy není potřeba starat se o uvolňování paměti.

Mezi hlavní výhody skriptovacích jazyků patří odstraněná nezbytnost program po každé změně kompilovat a z toho vyplývající snadná údržba, vývoj a správa kódu. Mezi hlavní nevýhody pak řadíme nižší rychlost oproti kompilovaným jazykům, větší paměťovou náročnost a v neposlední řadě také omezení týkající se přístupu do paměti, ovládání kontextových zařízení apod.

Mezi hlavní zástupce skriptovacích jazyků patří například Python, PHP, JavaScript, Perl, Tcl, Ruby nebo třeba také unixový shell. Ne všechny ze zmíněných zástupců jsou však vhodné pro tvorbu webových aplikací.

Mezi nejvíce používané skriptovací jazyky pro tvorbu webových stránek patří tyto tři:

PHP - Syntaxe tohoto jazyka je inspirována jazyky C, Java a Pascal a tento jazyk je možné přímo začlenit do HTML dokumentu, čímž je možné snadno vytvořit dynamicky generovaný obsah. Skripty v tomto jazyce fungují na mnoha operačních systémech a jazyk jako takový disponuje velkým množstvím dostupných knihoven, které rozšiřují jeho možnosti. Mezi jeho hlavní výhody patří jeho široká rozšířenost a podpora na poli hostingových služeb, mezi nevýhody pak absence debugovacího nástroje ve standardní distribuci, nejednotné pojmenovávání standardních funkcí a některé nekonzistence vyplývající z minulého vývoje jazyka.

Ruby - Jedná se o relativně mladý jazyk vzniklý a nejvíce rozšířený v Japonsku. Jde o běžný skriptovací jazyk, z čehož vyplývají také možnosti jeho použití. K jeho největšímu rozšíření na poli webových aplikací přispěl vznik velmi populárního frameworku Ruby on Rails, který funguje na bázi architektury Model-View-Controller. Mezi hlavní výhody jazyka Ruby patří jeho multiplatformnost, jednoduchá syntaxe a plná podpora objektově orientovaného programování. Mezi nevýhody pak kromě nedostatků existujících pro všechny skriptovací jazyky hraje roli také nedostatek české dokumentace a velké množství knihoven napsaných rovněž v Ruby.

Python - Je skriptovací jazyk, který disponuje obrovskou škálou možností a lze jej použít jak pro vývoj skriptů a webových aplikací, tak pro grafická uživatelská rozhraní a další aplikace. Jedná se o víceparadigmatický jazyk, což znamená, že je v něm možné programovat objektově, procedurálně a v omezené míře i funkcionálně. Jednou z nejdůležitějších vlastností je jeho jednoduchost, díky které je považován za nejvhodnější jazyk pro výuku programování. Výkon aplikací v Pythonu je na dobré úrovni, protože velká spousta základních knihoven je implementována v jazyce C (stejně jako hlavní interpretr jazyka), což jej v rychlosti zvýhodňuje.

S ohledem na výše popsaná fakta se jako nejvhodnější programovací jazyk pro tvorbu (nejen) webových aplikací jeví Python, který je rychlý a snadno použitelný k vývoji.

3.5 Asymetrická kryptografie

V následujících dvou kapitolách bychom se rádi zaměřili na teoretickou část asymetrické kryptografie a to především na její využití v šifrované komunikaci s použitím privátního a veřejného klíče a dále pak na systém certifikátů a certifikačních autorit.

3.5.1 Šifrovaná komunikace asymetrickou kryptografií

Kryptografie je z obecného pohledu věda zabývající se utajováním informací, resp. jejich převodem do podoby, která je bez speciálních znalostí nečitelná a nesrozumitelná. V moderní kryptografii se jako tato speciální znalost nejčastěji využívá vlastnictví nějakého klíče, hesla či speciálního zařízení, které je schopné informaci odtajnit.

Mezi dva nejznámější a v současné době i nejpoužívanější způsoby šifrování se řadí symetrická a asymetrická kryptografie. Zatímco symetrická kryptografie používá pro zašifrování i dešifrování

informace stejný klíč, u asymetrické kryptografie se jedná o dvojici klíčů, z nichž jeden je schopen informaci pouze zašifrovat, zatímco druhý je schopen tuto informaci pouze dešifrovat. Analogicky k příkladu s dveřmi bychom u symetrické kryptografie měli jeden klíč k jejich odemčení a stejný klíč k jejich opětovnému zamčení. U asymetrické kryptografie je vhodnější použít příklad s obálkou, kdy nám jeden klíč (lepidlo) obálku zalepí a druhý klíč (rozpouštědlo) obálku opět otevře.

Mezi nejznámější a nejrozšířenější algoritmy pro asymetrickou kryptografii se řadí algoritmus RSA, který využívá jednoduchého principu a při dostatečně dlouhém klíči je považován za bezpečný.

Princip RSA tkví v tom, že zatímco rozložit velké číslo na součin prvočísel (tzv. faktorizace) je velmi komplikovaná úloha, násobení dvou čísel je elementární úloha. Z velkého čísla N je v normálním čase prakticky nemožné zjistit faktorizací jeho prvočísla, protože neexistuje algoritmus faktorizace, který by pracoval v polynomiálním čase v závislosti na velikosti binárního zápisu čísla N.

Před zahájením samotné šifrované komunikace je nutné, aby si obě komunikující strany vytvořily pár klíčů, z nichž jeden bude veřejný a druhý privátní. Několika jednoduchými matematickými operacemi s použitím Eulerovy funkce dostaneme trojici čísel, z nichž je jedno označeno jako šifrovací exponent, druhé jako dešifrovací exponent a třetí jako modulo společné pro oba klíče. Dvojice šifrovacího exponentu a modula pak tvoří veřejný klíč a dvojice dešifrovacího exponentu a modula privátní klíč.

Na začátku šifrované komunikace si strany navzájem předají veřejné klíče, které mohou být přenášeny nezabezpečeným kanálem, zatímco privátní klíče si ponechají v tajnosti u sebe.

Při samotné komunikaci pak dochází k převodu tajné informace na číslo a jeho následné zašifrování veřejným klíčem protistrany. Naopak při přijetí zprávy použije příjemce svůj privátní klíč k dešifrování.

Algoritmus RSA je kromě šifrování vhodný i k podepisování zpráv. Podepsání zprávy funguje tak, že se nejdříve vypočte nějakým algoritmem kontrolní součet zprávy, který pak odesilatel dešifruje svým privátním klíčem a vzniklé číslo připojí jako součást zprávy. Příjemce si pak sám znovu vypočítá stejným algoritmem kontrolní součet zprávy a zároveň číslo ve zprávě zašifruje veřejným klíčem odesilatele. Pokud jsou tyto dva výsledky totožné, je zaručeno, že během přenosu nedošlo k modifikaci zprávy.

3.5.2 Certifikáty a systém certifikačních autorit

Digitální certifikát je veřejný šifrovací klíč, jak byl popsán v předchozí kapitole, který je digitálně podepsán a je vydáván certifikační autoritou. Certifikát obsahuje kromě informací nezbytných pro samotný proces šifrování také informace o jeho vlastníkovi a v neposlední řadě také o certifikační autoritě, která tento certifikát vystavila.

Tyto autority vystavují certifikáty v několika definovaných třídách, z nichž třída 1 je určena pro jednotlivce a podepisování emailové komunikace, třída 2 pro organizace s požadavkem na prověření identity, třída 3 pro servery a digitální podpisy s nezávislým potvrzením identity certifikační autoritou, třída 4 pro online transakce mezi společnostmi a poslední třída 5 pro soukromé subjekty či vládní organizace.

Certifikát samotný se pak vytváří tak, že certifikační autorita nejdříve ověří údaje o majiteli veřejného šifrovacího klíče, následně do něj doplní identifikační údaje a vše elektronicky podepíše, čímž zaručí pravost certifikátu a zamezí jeho následným modifikacím.

Při podepisování certifikátu certifikační autoritou se využívá tzv. přenosu důvěry: je-li certifikační

autorita důvěryhodná, jsou důvěryhodné i certifikáty, které vydala, a obsahují tedy jen pravdivé informace. Pro ověření podpisu jsou pak v počítači přítomny kořenové klíče certifikačních autorit, které umožňují verifikaci jimi vydaného certifikátu.

Certifikát pak může zaniknout buď po uplynutí doby jeho platnosti, nebo ještě dříve pomocí revokačního mechanismu. Ten funguje tak, že certifikační autorita vytváří seznam zrušených certifikátů (CRL) a jejich sériových čísel, který je běžně k dispozici a je možné se jím řídit při ověřování certifikátů a tedy nepřijmout certifikát, jehož sériové číslo se na CRL seznamu objeví. Další možností při ověření systému proti revokačním listům je přímý dotaz na online systém certifikační autority, který ovšem nemusí být všude podporován.

3.6 Škálování webových aplikací

S rostoucím počtem webových aplikací a jejich zvyšující se každodenní návštěvností se u mnohých z nich objevila potřeba jejich škálování, resp. přípravy aplikace do takového stavu, kdy tisícinásobná návštěvnost nebude znamenat logický či softwarový problém, bude se ale případně jednat jen o problém na straně hardware.

Ve škálování nejen webových aplikací existují dva přístupy: škálování vertikální a horizontální. Zatímco u vertikálního škálování se jedná o kvalitativní změnu stávajících prvků systému - např. rozšíření hardwarové základny serveru - u horizontálního se jedná o kvantitativní změnu a tedy o přidání nových prostředků například spuštění nové instance dané aplikace ve chvíli, kdy stávající instance nestíhá odbavit všechny požadavky.

Vzhledem k architektuře používané u současných webových aplikací a stále více se rozšiřujícím virtualizačním technologiím je možné efektivně využívat oba způsoby škálování - jak dynamické přidělování výpočetních zdrojů jednotlivým funkčním celkům, tak zvyšování počtu instancí jednotlivých komponent starajících se o běh celého systému - např. webové, aplikační či databázové servery.

Důležité je ovšem na tyto možnosti myslet již při návrhu infrastruktury a výběru softwarových součástí pro nově vznikající systém, ale zároveň nespoléhat čistě jen na ně a nezapomínat na optimalizaci aplikace samotné, která může ve finále potřebu škálování nejen oddálit, ale také tím ušetřit nemalé finanční prostředky za nákup nového vybavení.

4. Specifikace IS

Ve druhé části a v navazujících kapitolách se budeme zabývat specifikací celého informačního systému, dokumentací procesů, tvorbou diagramů a popisem chování jednotlivých jeho částí.

Budeme se zde také odkazovat na jednotlivé pasáže z první části práce pro porovnání současných a nově vznikajících procesů.

4.1 Uživatelé IS

Vznikající informační systém bude v závěru používat široká škála uživatelů, přičemž pro jednotlivé skupiny uživatelů bude informační systém disponovat různými možnostmi. Značná část informačního systému pak bude dostupná všem uživatelům, kteří se do něj zaregistrují a přihlásí.

Mezi hlavní skupiny uživatelů patří:

- běžní uživatelé superpočítačových služeb,
- hlavní řešitelé projektů,
- členové alokační a hodnotící komise.

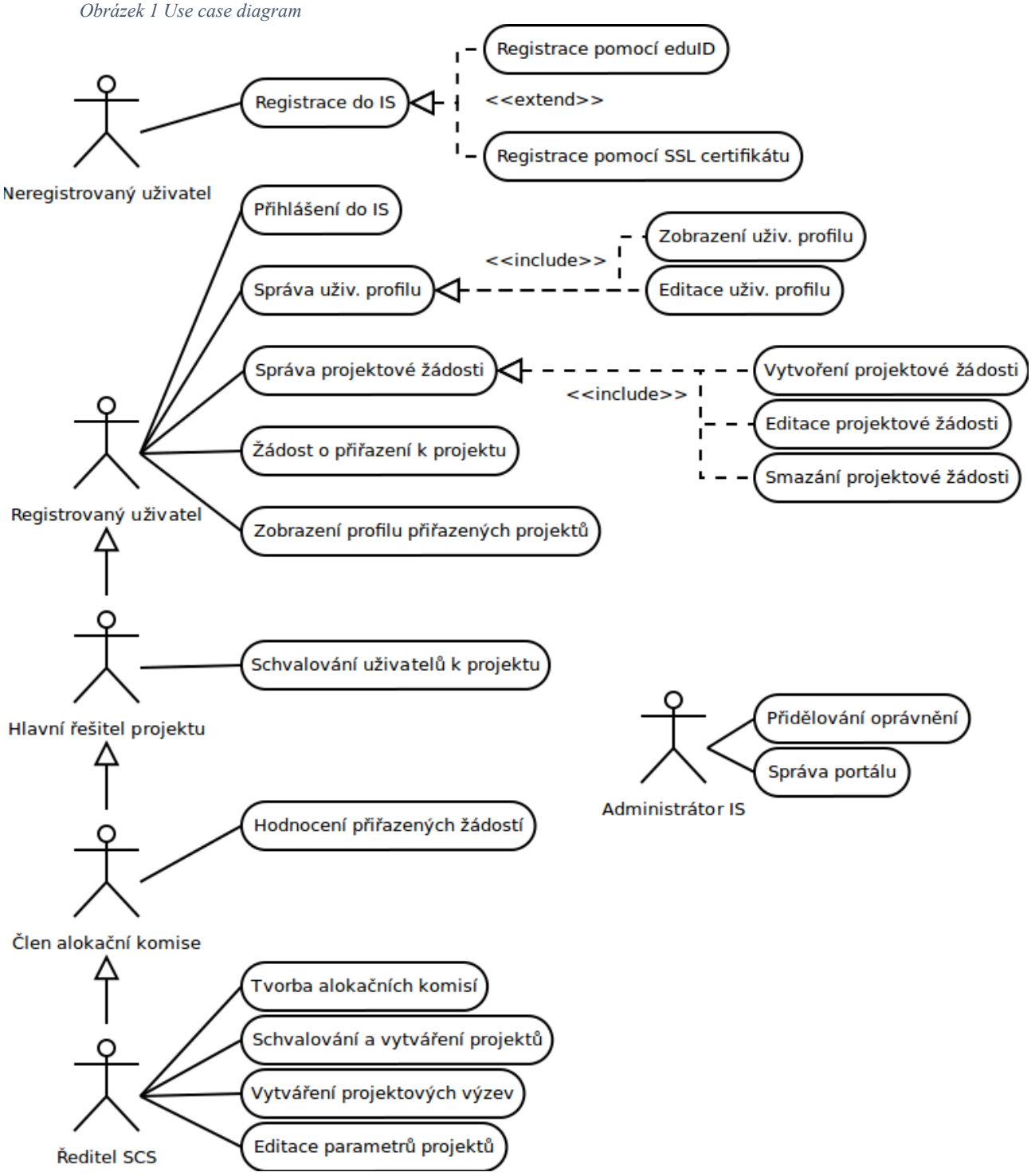
Stranou těchto skupin pak existují speciální případy:

- ředitel superpočítačových služeb, který má kromě práv výše zmíněných skupin také právo na tvorbu alokačních a hodnotících komisí a schvalování projektů, a tedy na přiřazování oprávnění ostatním uživatelům,
- administrátoři s plným oprávněním k celému IS.

4.1.1 Use Cases

V následujícím diagramu se pokusíme zachytit jednotlivé činnosti obsažené v IS a role uživatelů, kteří k těmto činnostem přistupují na základě svých oprávnění.

Obrázek 1 Use case diagram

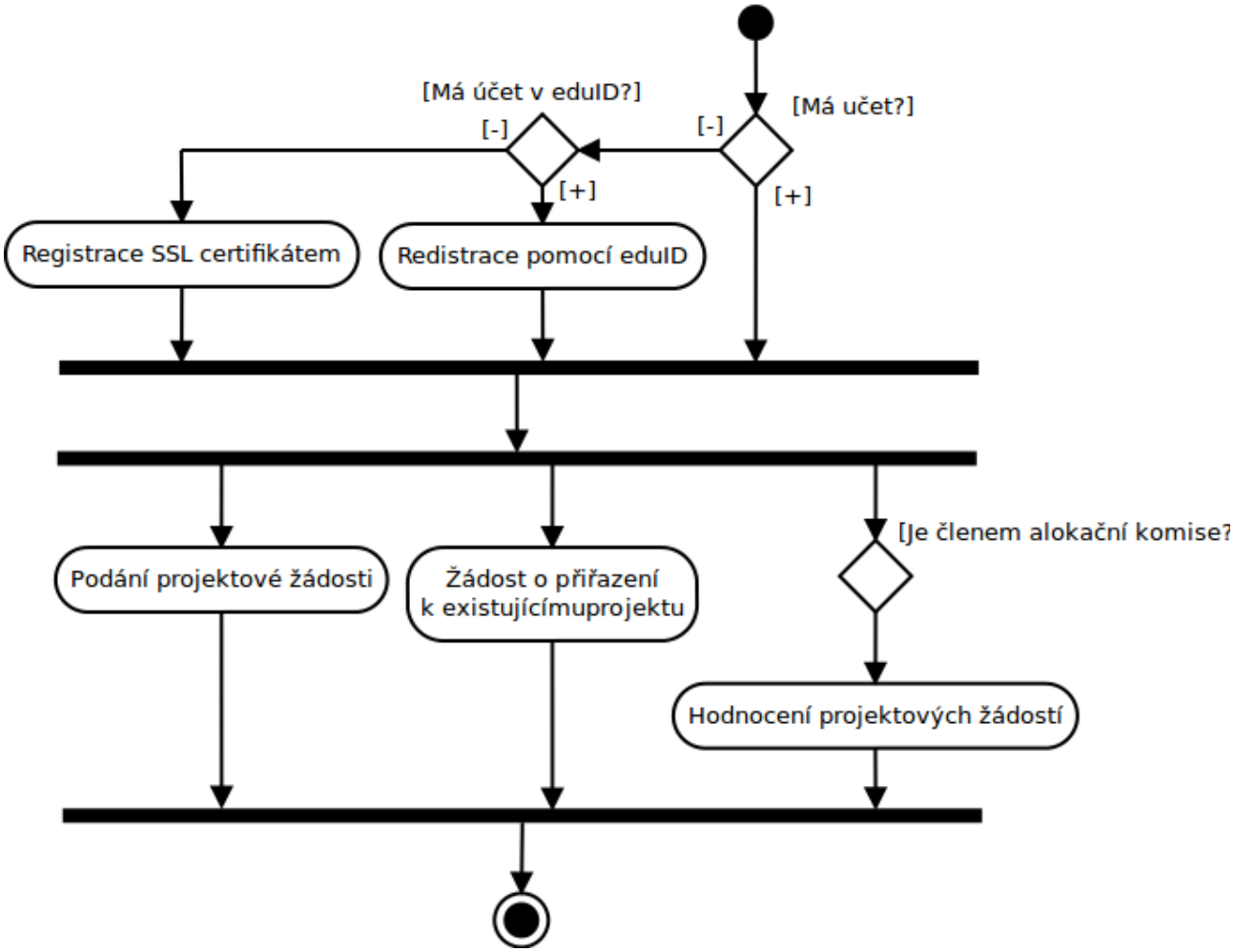


4.1.2 Workflows

Následující diagram aktivit popisuje zjednodušený proces, který bude muset absolvovat každý nově příchozí uživatel informačního systému.

Cílové činnosti jsou vybrány tak, aby popsaly nejčastější motivaci a možnosti nových uživatelů. V diagramu se zcela záměrně nevyskytuje část pro ředitele SCS, u kterého se počítá s členstvím v informačním systému od samého počátku.

Obrázek 2 Workflow diagram



4.2 Jednotlivé části IS

V následujících podkapitolách se budeme věnovat podrobnějšímu popisu jednotlivých částí informačního systému, jejich fungování, motivaci ke zvolenému způsobu implementace a v neposlední řadě bezpečnostním aspektům daného řešení.

4.2.1 Registrace

Registrace bude vstupním krokem pro všechny uživatele, kteří dosud nemají uživatelské konto v rámci superpočítačového centra. Důležitým faktem je, že ne všichni registrovaní uživatelé budou mít ve finále přístup k superpočítači, a tak je nezbytné uživatelům, kteří nejsou přiřazení k nějakému probíhajícímu projektu nebo nemají projekt vlastní, omezit práva na nezbytné minimum i s ohledem na zabezpečení informací uchovávaných v informačním systému.

V podstatě se bude jednat o možnost podat si projektovou žádost v některé z aktuálně otevřených výzev nebo o možnost požádat o přiřazení k existujícímu projektu. V prvním případě bude žadatel využívat informace, které jsou dostupné veřejně. V tom druhém bude muset znát identifikátor projektu, ke kterému se chce přihlásit, což už potvrdí jeho znalost, příp. možnost získat tuto informaci od hlavního řešitele daného projektu.

Ostatní funkce a informace obsažené v informačním systému budou pro čerstvě registrované uživatele nepřístupné.

4.2.1.1 Nároky na autorizaci a autentizaci uživatelů

V rámci služeb poskytovaných superpočítačovým centrem je nezbytné znát přesnou identitu uživatelů, kterým dané služby poskytujeme. Mimo jiné i z toho důvodu, že poskytované prostředky by se za jistých okolností daly použít k nelegální činnosti.

Uživatelé v současném procesu přidělování oprávnění souhlasí s dokumentem popisujícím pravidla chování v prostředí superpočítačového centra a užívání jeho prostředků, nicméně v případě jejich porušení, pokud bude odhaleno, je nezbytné mít jistotu, že známe přesnou identitu uživatele, který se pod daným účtem nevhodného chování dopustil.

Se znalostmi ze současného provozu superpočítačového centra jsme se rozhodli pro implementaci dvou možností registrace, z nichž každá nám umožní ověřit přesnou identitu uživatelů a informace o ní přenést do databáze v informačním systému.

Těmito možnostmi jsou:

- registrace pomocí osobního SSL certifikátu,
- registrace pomocí identity v eduID.

Registrace osobním SSL certifikátem, který je vydán důvěryhodnou certifikační autoritou a obsahuje nezbytné údaje o uživateli a jeho organizaci poslouží k prokázání jeho identity a získání nezbytných informací z důvěryhodného zdroje.

EduID je česká akademická federace identit, která na akademické půdě sdružuje poskytovatele identit, kterými jsou zpravidla domovské organizace jednotlivých uživatelů. Zapojením se do eduID dává organizace svým uživatelům možnost využívat různých služeb poskytovatelů, kteří mají eduID implementováno jako ověřovací mechanismus, používat jednotné přihlášení, a to vše při zachování pouze jednoho uživatelského účtu v rámci své domovské organizace.

Tyto dvě možnosti by dle současných zkušeností měly zahrnout většinu příchozích uživatelů, pro které by použití jedné či druhé metody nemělo znamenat žádné výrazné komplikace. Pro uživatele, kteří nemají v rámci domovské organizace přístup k SSL certifikátům či identitě v eduID, je zde stále možnost využít některou důvěryhodnou externí certifikační autoritu a osobní SSL certifikát si zakoupit.

Pokud by se i přes velké pokrytí objevily potíže s registrací, bude muset technická podpora superpočítačového centra řešit tyto případy individuálně. V minulosti se jich řešilo několik a zpravidla byla jejich příčina v nedostatečné počítačové gramotnosti uživatelů. Jako jedno z možných řešení se nabízí osobní návštěva uživatele a prokázání své identity platným dokladem totožnosti.

4.2.1.2 SSL Certifikátem

Zmiňovaná registrace pomocí SSL probíhá ve dvou rovinách:

- pokázání identity serveru a zabezpečení komunikace,
- prokázání identity uživatele.

První zmiňovaný proces je dnes již velmi běžný a vyžaduje od správce serverů zařízení důvěryhodného certifikátu pro konkrétní doménové jméno a jeho nasazení na webový server. Webový prohlížeč pak reflektuje jednak na poskytnutou identitu poskytovatele obsahu a také na zabezpečení komunikace. V dnešní době je toto již standardem a SSL nad HTTP protokolem používá čím dál více poskytovatelů obsahu a internetových aplikací.

Druhý a v našem případě podstatnější mechanismus vyžaduje od uživatele v prohlížeči nainstalovaný důvěryhodný osobní certifikát. Konfigurace webového serveru není v tomto případě nijak výrazně složitější.

Několika konfiguračními direktivami se webovému serveru předá nastavení URL adresy, na které má být klientský certifikát vyžadován, soubor obsahující jednotlivé certifikáty důvěryhodných certifikačních autorit a také tzv. hloubku zkoumání důvěry. Poslední zmíněný termín znamená, jak hluboko se bude v řetězci důvěry mezi certifikáty postupovat, aby došlo k nalezení podpisu důvěryhodnou certifikační autoritou. Hodnota 1 v tomto případě znamená, že klientský certifikát musí být důvěryhodnou certifikační autoritou podepsán přímo.

Takto nakonfigurovaný webserver bude při přístupu na předem určenou adresu vyžadovat klientský certifikát a při jeho poskytnutí dojde k ověření. Pokud bude certifikát uznán, zobrazí se uživateli obsah uložený na dané adrese.

Pro naše využití je zde ještě jedna důležitá konfigurační direktiva, která webserveru nastaví, že má informace získané z certifikátu poskytnout aplikačnímu serveru ve formě proměnných prostředí, díky čemuž máme možnost s těmito informacemi dále pracovat.

Jakmile si tedy uživatel zvolí způsob registrace pomocí SSL certifikátu, dostane se na adresu, kde je webserverem vyžadován klientský certifikát. Stále v rámci webserveru dojde k jeho ověření a pokud je vše v pořádku, zobrazí se uživateli registrační formulář. Díky zpracování informací z webserveru na aplikační úrovní dokáže informační systém ověřit, zda tento certifikát již nebyl použit dříve a pokud ne, vyplní do registračního formuláře nezbytné informace, které se podařilo z certifikátu získat. Tato formulářová pole jsou nastavena tak, aby je uživatel nemohl měnit a aby se tím zabránilo podvržení identity.

Uživatel v tomto kroku doplní další nezbytné údaje, které z certifikátu získat nelze, a následně registrační formulář odešle.

V dalším kroku dojde znovu k získání informací z certifikátu a ověření, zda se informace z něj shodují s informacemi z odeslaného formuláře. Toto je ochrana proti případnému útočníkovi, který by

podvrhl data odeslaná ve formuláři, změnil by si tímto identitu a snažil by se provést registraci s rozdílnými údaji oproti obsahu SSL certifikátu.

4.2.1.3 EduID

Ve federaci eduID [1] je možné vystupovat ve dvou rolích.

- jako identity provider,
- jako service provider.

Identity providerem jsou nejčastěji organizace, které provozují služby umožňující autentizaci uživatelů a distribuci informací o nich. Tímto umožní využívat služby poskytované service providery.

Service provider je správce služby, který se rozhodl do své aplikace implementovat možnost využití eduID a tím umožnil její využití širokému spektru uživatelů, jejichž organizace jsou do federace zapojeny.

Možnost registrace pomocí eduID by měla pokrýt značné procento budoucích uživatelů, protože se předpokládá, že většina uživatelů superpočítačových služeb bude pocházet z akademické sféry, která je systémem eduID dobře pokryta.

Základem komunikace, při které dojde k využití eduID, je nejdříve přístup na stránku service providera. V našem případě se jedná o registrační stranu, kde si uživatel může zvolit registraci pomocí eduID.

Po této volbě je přesměrován na stránku s výběrem domácí instituce, tzv. "Where Are You From" neboli WAYF, kde najde svou domácí organizaci a zvolí ji. Následně je přesměrován na důvěrně známé stránky své domovské organizace, kde provede ověření své identity, jak je zvyklý. Ověření identity na straně identity providera může být implementováno mnoha různými způsoby a je na celém procesu nezávislé.

Po ověření identity je uživatel přesměrován zpět na stránku service providera, který mimo informací o autentizaci může dle potřeby obdržet i další detailní informace o uživateli. V našem případě se po úspěšném přihlášení uživateli předvyplní registrační formulář informacemi získanými od identity providera. Přičemž tyto informace nebude moci uživatel změnit. Tím dojde k ověření identity uživatele a k jejímu uložení v důvěryhodné podobě do databáze informačního systému.

I zde dokáže informační systém na aplikační úrovni ověřit pomocí kontrolního součtu, zda již nebylo konto ve federaci použito k registraci do portálu, a tím zamezit duplicitám.

Za většinou z tohoto procesu stojí projekt *Shibboleth* [2] amerického Internetu2, který se pomocí open source nástrojů snaží poskytnout prostředky pro SSO přístup ke stránkám napříč organizacemi. Shibboleth přitom musí být správně nakonfigurován a úzce spolupracuje s webovým serverem, který si jeho použití dokáže na jistých URL adresách vynutit stejně jako uživatelský certifikát v případě registrace pomocí SSL.

4.2.2 Přihlášení

Přihlášení je klasickým prostředkem při přístupu k informačním systémům. Jeho prostřednictvím uživatel prokáže znalost nezbytných informací pro přístup k datům.

V informačním systému pro superpočítačové centrum jsou v pozadí přihlášení dva mechanismy,

které samotné přihlášení zajišťují:

- přihlášení existujícím účtem v superpočítačovém centru,
- přihlášení lokálně vytvořeným účtem v portálu.

4.2.2.1 Existujícím účtem v SCC

Uživatelé, kteří již mají v superpočítačovém centru konto, mohou s jeho pomocí přistupovat i k tomuto informačnímu systému. Tato možnost existuje proto, že i stávající uživatelé mohou často mít stejné požadavky jako nově příchozí. Například se může jednat o možnost podání další projektové žádosti, přiřazení k dalšímu projektu apod.

Správa uživatelských účtů je prováděna v LDAPu a informace o uživatelích jsou používány napříč celým superpočítačovým centrem a službami, které poskytuje.

4.2.2.2 Novým účtem v portálu

Čerstvě registrovaný uživatel využije pro přihlášení nově vzniklé konto lokálně uložené v informačním systému.

Uživatelé můžeme u tohoto typu přihlášení rozdělit do dvou kategorií:

- uživatel, který má ambici používat superpočítačové služby,
- uživatel, který bude využívat jen služeb informačního systému.

První skupina se registruje do portálu s ambicí, že se jeho prostřednictvím dostane jejich účet do centrálního registru a budou tak moci využívat superpočítačových služeb. To lze docílit při schválení projektové žádosti nebo třeba při schválení přiřazení uživatele k existujícímu projektu. V takovém případě se konto uživatele přesune z lokálně uloženého v informačním systému do centrálního registru v LDAPu a uživatel se tak stane plnohodnotným.

Druhá skupina se v portálu registruje jen pro získání lokálního konta. To mohou být například členové hodnotící nebo alokační komise. Tito pracovníci přistupují lokálním kontem do informačního systému za jistým účelem, který v jejich případě nepovede k získání konta na superpočítači, a tak jej budou i nadále využívat k přihlášení do lokálního účtu.

4.2.3 Podávání projektových žádosti

Podávání projektových žádosti není samo o sobě nijak komplikované, právě pro to, aby jej zvládl kdokoli i z řad méně počítačově gramotných uživatelů informačního systému, ale zároveň se také řídí celou řadou pravidel a omezení.

V prvé řadě je třeba mít k podání projektové žádosti otevřenou výzvu. Výzvy se otevírají ve více či méně pravidelných intervalech a dělí se do několika skupin:

- interní grantová soutěž - vypisovaná pravidelně 4x ročně pro zaměstnance IT4Innovations a jejich spolupracovníky;
- veřejná grantová soutěž - vypisovaná pravidelně 2x ročně pro zaměstnance jiných výzkumných, vědeckých a vzdělávacích organizací, než je IT4Innovations;
- přidělení výpočetního času rozhodnutím ředitelství - žádost lze podat kdykoliv; jedná se o nepravidelné přidělování výpočetního času na základě posouzení ředitelství IT4Innovations.

Z výše uvedeného vyplývá, že pokud chce uživatel využít jedné z pravidelně vypisovaných výzev, je nutné dodržet lhůtu pro podání žádosti.

Při podávání žádosti v informačním systému jsou uživateli nabídnuty možnosti pro výběr konkrétní výzvy s tím, že seznam reflektuje jejich současný stav. Pokud zároveň dojde k podání žádosti do jedné z pravidelně vypisovaných výzev, je u této žádosti reflektován termín pro její dokončení, po jehož uplynutí nemůže již uživatel žádost dále upravovat a ta je tedy postoupena dále do schvalovacího procesu.

Výše zmíněné se netýká poslední zmíněné kategorii žádostí, o jejichž schvalování může ředitelství rozhodnout kdykoli a ani jejich úpravy tak nejsou časově omezeny.

Mimo výzvy, do které žadatel svou žádost zařadí, musí doplnit i následující údaje:

- název projektu,
- požadovaný počet jádrohodin - minimum je 384, maximum 4 000 000,
- speciální požadavky na typ výpočetních prostředků,
- vědní obor, kam tento projekt spadá,
- přihlašovací jméno - needitovatelný údaj, pouze informačního charakteru,
- jméno a příjmení - editovatelný údaj získán při registraci,
- oslovení pro účely následující emailové komunikace,
- domovská organizace uživatele,
- oddělení v organizaci - nepovinný údaj,
- abstrakt vhodný pro publikaci na webových stránkách a v jiných médiích. Maximální délka je 1500 znaků,
- příloha ve formě obsáhlejšího slovního popisu žádosti.

Po správném vyplnění je uživateli jeho žádost uložena do systému i s informací, dokdy je možné ji editovat. Po uplynutí tohoto data ji uživatel stále vidí, ale již bez možnosti editace.

4.2.4 Schvalování žádostí

Schvalování žádostí je jedna z nejkomplexnějších úloh v informačním systému, a především vzhledem k počtu zainteresovaných pracovníků a také počtu kroků, které stojí mezi žádostí a finálním projektem v superpočítačovém centru.

Schvalování se skládá z několika kroků:

1. podání žádosti uživatelem (popsáno v předchozí kapitole),
2. vytvoření hodnotící komise,
3. hodnocení hodnotící komisí,
4. zpracování výsledků hodnotící komise alokační komisí,
5. určení finálního rozhodnutí o budoucnosti projektu a jeho konečné alokaci.

Nyní se na jednotlivé kroky podíváme podrobněji.

4.2.4.1 Tvorba alokační komise

Alokační komise je hlavní orgán, který rozhoduje o konečném schválení projektů a určení jejich finální alokace.

Alokační komisi vytváří ředitel superpočítačového centra a členové této komise se v čase mění jen zřídka.

4.2.4.2 Fungování alokační komise

Každý člen alokační komise má možnost prohlížet si žádosti včetně příloh, u žádostí sestavovat hodnotící komisi, navrhovat finální alokaci projektu a sledovat hodnocení jednotlivých žádostí hodnotící komisí.

Celý proces pak vypadá tak, že se na začátku hodnocení vytvoří každému projektu hodnotící komise, následně se počká, až jsou jednotlivé žádosti ohodnoceny a na základě toho dojde ke schválení či zamítnutí projektu. V případě schválení je také určena finální alokace, která může a nemusí být stejná, jako byla požadovaná alokace.

4.2.4.3 Tvorba hodnotící komise

Každý člen alokační komise má právo u jednotlivých žádostí upravit složení hodnotící komise.

Hodnotící komise nejčastěji obsahuje dva na sobě nezávislé odborníky z oblasti, které se věnuje žadatel v žádosti o svůj projekt.

Skupina členů, ze kterých je možné hodnotící komisi sestavit, je předem dána a ředitel má právo tuto množinu upravovat. Členem hodnotící komise má možnost být kdokoli, ale z bezpečnostních a praktických důvodů je seznam možných členů omezen.

Tato skupina osob, je rovněž přiřazena ke konkrétnímu druhu výzvy. Jakmile dostane uživatel nějakou žádost k hodnocení, má možnost nahlédnout do všech žádostí ze stejné výzvy, avšak hodnocení může psát jen pro jednu z nich.

4.2.4.4 Fungování hodnotící komise

Jakmile je libovolný uživatel označen za člena hodnotící komise pro daný typ výzvy, objeví se mu v informačním systému nová část, která obsahuje všechny aktuálně podané žádosti pro danou výzvu, ke které byl přiřazen. Hodnotitel si může tyto žádosti procházet a to včetně projektových příloh.

Člen alokační komise pak může takového člena přiřazeného k danému typu výzvy dále přiřadit ke konkrétní žádosti, čímž tomuto členovi hodnotící komise umožní k žádosti přidat vlastní slovní hodnocení ve zmíněných aspektech a ke každému z nich také bodové ohodnocení.

Člen hodnotící komise je po přihlášení do informačního systému upozorněn na přítomnosti projektových žádostí, které čekají na jeho hodnocení.

Komise má po jejím ustanovení určitý čas na to, aby k přiřazené žádosti doplnila hodnocení. Žádosti se hodnotí v několika ohledech, přičemž v každém z nich se provádí hodnocení slovní a následně také číselné, dle zadaných kritérií. Perspektivy, ze kterých se na žádost při jejím hodnocení nahlíží, jsou následující:

- odborná připravenost (číselné hodnocení v intervalu <1,5>),
- připravenost pro výpočet (číselné hodnocení v intervalu <1,5>),

- socioekonomický dopad (číselné hodnocení v intervalu <0,5>).

Členové hodnotící komise mají až do zadaného termínu možnost své hodnocení libovolně upravovat.

Takto vložené hodnocení je viditelné i uživateli, který žádost podal, ale pouze v anonymizované formě.

4.2.4.5 Vytvoření projektů

Jakmile je u všech projektových žádostí zadáno hodnocení přiřazenými členy hodnotící komise a došlo ke shodě členů alokační komise na finální alokaci, je možné z této žádosti vytvořit nový projekt. Vytvoření nového projektu se provede jako umístění projektové žádosti do fronty pro zpracování navazujícím informačním systémem.

Vytvoření projektu fakticky probíhá v jiném informačním systému, kde do této doby vytvářeli projekty ručně administrátoři superpočítačového centra. Z tohoto informačního systému pak čerpají informace další napojené systémy včetně plánovače úloh na superpočítači.

Po vytvoření projektu se žádost z námi popisovaného portálu odstraní a zůstane pouze v archivu. Tímto se zamezí zvyšování počtu žádostí v jednotlivých sekcích portálu a tím snížení jeho přehlednosti, ale všechny žádosti zůstanou uloženy stranou pro případnou další kontrolu.

Pokud dojde k zamítnutí žádosti, je možné ji buď smazat bez archivace, nebo ji archivovat bez vytvoření projektu.

4.2.5 Správa vlastních projektů

Následující podkapitoly přiblíží specifikaci, jaké možnosti nabízí informační systém uživatelům, kteří ze své žádosti získali finální projekt. Popis bude rozdělen do dvou samostatných částí, kdy každá z nich se bude zabývat pohledem uživatele v jiné rolí. Jedna kapitola bude určena pro fungování informačního systému z pohledu hlavního řešitele projektu. Druhá pak z pohledu běžného uživatele superpočítačových služeb.

4.2.5.1 Z pohledu PI

Hlavní řešitelé projektů získají díky tomuto informačnímu systému lepší přehled o svých projektech a jednodušší možnosti správy projektů.

Mezi hlavní proces, který by se měl v rámci informačního systému zlepšit, patří přiřazování uživatelů, které doposud probíhalo ručně a bylo komunikováno skrze systém podpory uživatelů.

Mezi další užitečné funkce řadíme možnost nahrávání publikací, odevzdávání závěrečných zpráv a odesílání standardizovaných žádostí, jako je například žádost o prodloužení projektu.

4.2.5.1.1 Přiřazování uživatelů

Hlavní funkcionalitou této sekce bude ovšem zmíněné přiřazování uživatelů k projektu, kde je aktuálně přihlášený uživatel hlavním řešitelem.

Hlavní řešitel bude mít u každého svého projektu možnost přiřadit libovolného uživatele k tomuto projektu a tím mu umožnit využít výpočetní čas alokovaný v rámci tohoto projektu. Přiřazení bude možné provést buď přímo hlavním řešitelem, kdy si hlavní řešitel vybere uživatele a toho následně přiřadí, nebo

pomocí schválení žádosti, kterou podá uživatel ve stejném informačním systému.

U první zmíněné možnosti je ovšem nezbytnou podmínkou, aby měl přiřazovaný uživatel již existující konto, zatímco při schvalování žádosti od uživatele a v případě neexistence účtu, bude uživateli účet založen automaticky při schválení.

4.2.5.2 Z pohledu uživatele

Uživatelé budou moci v informačním systému dohledat detailní informace k projektům, u nichž jsou přiřazeni, ale nebudou mít stejná oprávnění jako hlavní řešitel projektu.

Tato sekce jednak umožní uživateli náhled na projekty, ke kterým je přiřazen a jejichž prostředky může využívat, ale také si požádat o přiřazení k jednotlivým projektům.

Tato část se z pohledu uživatele jeví jako jedna z nejdůležitějších. Pokud totiž nemá uživatel ambice pro podávání projektové žádosti, budou jeho první kroky po registraci směřovat do této sekce, kde díky žádosti o přiřazení k některému z projektů získá uživatelské konto.

4.2.5.2.1 Požadavek k přiřazení k projektu

Pro podání žádosti bude uživatel muset vyplnit jednoduchý formulář, který se bude skládat jen z identifikátoru projektu, do kterého si uživatel přeje být přiřazen.

Po odeslání této žádosti se hlavnímu řešiteli zadaného projektu zobrazí možnost jejího schválení s informací, zda si o přiřazení požádal existující uživatel superpočítače nebo nově příchozí uživatel, který disponuje pouze registrací v informačním systému.

Jakmile bude žádost schválena, uživateli se odešle informační email, který jej informuje o schválení či zamítnutí žádosti a ihned se mu také v této sekci objeví informace o nově přiřazeném projektu.

4.2.5.2.2 Získání uživatelského účtu

Jak již bylo popsáno výše, uživateli, který podá žádost o přiřazení k projektu a dosud nedisponuje uživatelským účtem na superpočítači, bude při schválení jeho žádosti uživatelské konto vytvořeno a přiřazeno k projektu, jehož hlavní řešitel vytvoření konta inicializoval schválením žádosti o přiřazení.

Díky komplexnímu systému registrace se sběrem všech potřebných údajů nebude muset uživatel v tomto kroku již vyplňovat žádné dodatečné informace. Stávající informace se z uživatelova lokálního konta v informačním systému přesunou do centrálního systému správy identit, ve kterém se mu vytvoří a přiřadí chybějící atributy.

Takto vytvořené konto disponující potřebnými informacemi a heslem, které uživatel používá pro přístup do informačního systému, bude v tuto chvíli moci využívat i pro další navazující systémy a portály superpočítačového centra.

Pro přístup k superpočítači samotnému skrze SSH bude ovšem vyžadován ještě jeden nezbytný krok: nahrání uživatelova veřejného SSH klíče do systému, odkud se přenese do centrální databáze identit. Protože se jedná o změnu citlivých údajů uživatelova účtu, bude muset uživatel před jejím provedením projít ověřením jedním ze svou způsobů, které se používají při registraci.

Jednou z původně zvažovaných možností bylo generování klíčů pro uživatele přímo v jeho prohlížeči pomocí Javascriptu, čímž by se zajistilo, že privátní část klíče nebude nikdy přenášena po síti. Toto řešení jsme ale nakonec zavrhli hned ze dvou důvodů. Jednak by zde mohl na starších počítačích

a různých prohlížečích vzniknout problém s nekompatibilitou a pomalým generováním klíčů, a také se mi nepodařilo naimplementovat algoritmus, který by klíče dokázal generovat již zaheslované.

Z výše zmíněných důvodů bude nejsnazší, když si uživatel jednoduše vygeneruje pár klíčů na svém počítači a do systému nahraje jen veřejnou část z této dvojice. V neposlední řadě to ulehčí situaci těm uživatelům, kteří již SSH klíče využívají a budou tak moci použít stávající klíč.

4.3 Zabezpečení

Na bezpečnost webových aplikací se v dnešní době klade velký důraz a tento důraz je o to patrnější tam, kde je webová aplikace rozhraním k přístupu k citlivým údajům.

V této kapitole bych rád zmínil některé známé útoky na webové aplikace a princip, jak se v rámci informačního systému bránit proti jejich využití.

Konkrétně půjde o následující hrozby:

- SQL injection,
- XSS,
- Krádež session,
- MITM útok.

4.3.1 SQL injection

SQL injection je jeden z nejčastějších útoků na webové aplikace. Útok využívá toho, že jsou vstupy od uživatele bez zpracování použity k vytvoření SQL metody a tato metoda je pak spuštěna nad databází. Útočník tak může do vstupu pro webovou aplikaci vložit znaky, které vyvolají v SQL metodě chybu, resp. upraví její chování tak, aby získal přístup i k datům, které jsou běžně nedostupné.

Jako ochranu proti takovým útokům používáme při tvorbě SQL metod v informačním systému šablonovací jazyk DTML, pomocí kterého vkládáme parametry do SQL metod. U takto vložených parametrů můžeme mít jistotu, že došlo ke zpracování všech nežádoucích složek parametru a výsledná SQL metoda je tak bez problému použitelná.

Velká výhoda tkví v tom, že ošetřování parametrů probíhá právě při přípravě SQL metody a nezáleží na tom, odkud se parametr vzal. Není tedy nutné ošetřovat parametry ještě před jejich předáním do SQL metody.

4.3.2 XSS

XSS, neboli Cross-site scripting, je v poslední době velice populární a nenáročná forma útoku, která jako prostředek využívá zpracování vstupu od uživatele bez ošetření a jeho následné vložení do stránky.

Rozlišujeme celkem tři typy XSS:

1. lokální,
2. reflected,
3. persistent.

První dva útoky jsou si velmi podobné. První zmiňovaný využije zpracování neošetřeného vstupu (například z URL) v javascriptu na straně klienta. Útok tak umožní odesláním podvodné URL adresy ze strany klienta spuštění kódu na jeho straně. Tento typ útoku je snadno odhalitelný právě kontrolou URL adresy.

Druhý typ je prvnímu velice podobný. Rozdíl mezi nimi spočívá v tom, že u reflected útoku nedochází ke zpracování neošetřeného vstupu na straně klienta v javascriptu, ale na straně serveru, resp. aplikace samotné. Aplikace pak reflektuje neplatný vstup do těla stránky a umožní tak spustit podvodný kód na straně klienta, který na postiženou URL přistoupí.

Poslední typ je ze všech zmíněných nejzávažnější. Vektor jeho útoku předpokládá uložení neošetřeného vstupu do paměti aplikace a tím trvalou modifikaci webového obsahu. Příkladem může být opět volání URL, pomocí které se do databáze aplikace uloží jméno zákazníka. Když v tomto volání podvrhneme jméno zákazníka za kus JS kódu, provede se tento JS u každého klienta, který navštíví stránku, v níž je jméno zákazníka zobrazeno.

Jako aktivní ochranu proti tomuto typu útoku jsem se při vývoji zaměřil na ošetřování vstupů od uživatelů informačního systému, přetypovávání proměnných na jejich očekávaný typ a v neposlední řadě na minimum reflektovaných informací z URL.

Jako další stupeň ochrany informačního systému proti tomuto typu útoku slouží vestavěné funkce šablonovacího systému, které nedovolí v rámci dat získaných z databáze či jiného zdroje vložit do těla stránky další HTML kód. A tedy i v případě, kdy se do databáze omylem dostane škodlivý kód, šablonovací systém jeho přetypováním na HTML entity a zobrazením v čitelné podobě zabrání jeho vykonání.

4.3.3 Krádež session

Krádež sezení a tedy získání přístupu k aplikaci jiného přihlášeného uživatele je rovněž velmi populárním útokem a úzce souvisí s předchozím zmiňovaným XSS útokem.

Protože je HTTP protokol bezstavový, ale dnešní webové aplikace potřebují přenášet alespoň minimum informací o stavu komunikace s daným klientem, vznikla metoda sezení, kdy je uživatel jednoznačně identifikován pomocí náhodně generovaného identifikátoru. Tento identifikátor se po přihlášení uloží na straně uživatele a přenáší se při každém požadavku na aplikaci. Aplikace tak dokáže uživatele identifikovat jako přihlášeného a poskytnout mu odpovídající obsah.

Identifikátor sezení se přednáší buď prostřednictvím URL, což je ta méně bezpečná varianta, nebo se ukládá do *cookie*.

Právě XSS útok se používá ke krádeži identifikátoru přihlášeného uživatele tak, že se do napadené stránky vloží prostřednictvím neošetřeného vstupu JavaScriptový kód, který přečte *cookies* na straně uživatele a nějakým způsobem je dopraví k útočníkovi. Útočník si pak identifikátor nastaví ve svém prohlížeči a při přístupu do stejné aplikace bude vystupovat v roli napadeného uživatele.

Jako ochranu proti tomuto útoku jsme již zmínili ochrany proti XSS útokům. Dále je jako jeden z hlavních prvků ochrany identifikátoru sezení nastaven pro danou *cookie* parametr httpOnly, který zajistí, že informace uložená v dané *cookie* je čitelná pouze skrze HTTP protokol a JavaScript na straně klienta se k ní tedy nedostane ani v případě úspěšného XSS útoku.

4.3.4 MITM útok

Útok zvaný Man In The Middle není již tak častý jako výše zmíněné útoky, a to především z toho důvodu, že ke svému provedení vyžaduje od útočníka, aby se dostal ke komunikačnímu kanálu, kterým komunikace mezi serverem a klientem probíhá. U předešlých útoků nebylo nic takového potřeba, a tak mohou cílit na větší skupinu uživatelů.

Pokud už se útočník při MITM dostane k přenosovému kanálu a tento kanál je nešifrovaný, může nejen jednoduše odposlouchávat zprávy jdoucí oběma směry, ale také je může dle potřeby zachytávat a měnit, aniž by kterákoli strana tuto manipulaci odhalila.

Zabránit tomuto útoku se dá využitím asymetrické kryptografie, při které si jednotlivé strany na začátku komunikace vymění veřejné klíče, které slouží k dešifrování zpráv, a následná komunikace již probíhá šifrovaně. Potíž je ovšem v tom, že když bude útočník přítomen u začátku takové komunikace, může odchytit i výměnu veřejných klíčů, podvrhnout je a tím se dostat i do šifrované komunikace.

Jako ochranu proti odposlouchávání informací na jejich cestě mezi uživatelem a serverem jsme celý portál zabezpečili použitím protokolu HTTPS, který využívá certifikát vydaný platnou certifikační autoritou a je tedy posuzován jako platný ve všech prohlížečích a operačních systémech.

I zde je možnost provedení MITM útoku, při kterém by útočník certifikát serveru podvrhl, ale v tom případě by byl certifikát označen jako neplatný a od uživatele by vyžadoval explicitní potvrzení jeho použití. Další možností jak ochranu obejít je podvrhnout uživateli DNS nebo ARP záznamy a přesměrovat jej skrze proxy, která bude sice se serverem komunikovat šifrovaně a s platným certifikátem, ale uživateli zprostředkuje nešifrované spojení.

Obě zmíněné možnosti napadení zvoleného zabezpečení vyžadují explicitní povolení uživatele pro použití nedůvěryhodného způsobu komunikace nebo jeho sníženou obezřetnost.

5. Ekosystém aplikací SCC

Jedním z hlavních požadavků na informační systém bylo jeho zapojení do existující infrastruktury informačních a jiných systémů, které jsou v provozu superpočítačového centra aktivně využívány, a na jejichž dostupnosti a bezchybnému provozu závisí (ne)možnost uživatelů plnohodnotně využívat přidělený výpočetní čas.

Do jisté míry tím byly předem určeny technologie, které se při vývoji tohoto nového informačního systému použijí. Hlavní motivací k setrvání v zaběhnutých technologických kolejích byla možnost správy nového informačního systému stávajícími administrátory, kteří již mají zkušenosti s fungující infrastrukturou superpočítačového centra a u kterých tím odpadne nutnost nákladného a časově náročného školení.

5.1 Navazující portály

SCCS portál je historicky první portál, který vznikl pro podporu provozu superpočítačového centra.

SCCS portál je postaven na stejných technologiích, jako nově vznikající informační systém, ale jeho logické začlenění mezi ostatní prvky existujícího ekosystému je spíše zaměřeno na administraci informací důležitých pro provoz superpočítačového centra. Mezi hlavní spravované informace patří informace o jednotlivých projektech, jejich parametrech, uživatelích do projektů zapojených a zdrojích, které mají jednotlivé projekty k dispozici. Mimo vlastní databázi slouží tento informační systém také jako read-only rozhraní pro zobrazení informací o uživatelích z LDAP serveru.

SCCS portál obsahuje také API, které je použité pro interní potřeby superpočítačového centra a které umožňuje v reálném čase získávat informace o zdrojích dostupných pro jednotlivé projekty. Toto API je využité přímo v plánovači úloh, který na základě informací získaných z tohoto API rozhodne o spuštění či zamítnutí výpočetní úlohy daného uživatele v rámci daného projektu.

Vzhledem k popsanému účelu není v tomto portále nutné řešit tak jemné rozdělení přístupových oprávnění, protože uživatelé, kteří SCCS portál nejčastěji používají, jsou zpravidla administrátoři celého superpočítačového centra a z tohoto titulu mají i maximální možná oprávnění.

Nově vznikající informační systém s SCCS portálem úzce spolupracuje především ve chvíli, kdy v novém informačním systému dojde ke schválení žádosti o přidělení výpočetního času. Po takovém schválení žádosti se automaticky přenesou všechny informace z jednoho portálu do druhého, v SCCS portálu se vytvoří nový projekt s určitými parametry, které jsou buď výchozí pro každý projekt, nebo byly obsaženy v konkrétní žádosti.

Další úroveň spolupráce spočívá ve sdílení informací mezi databázemi těchto portálů tak, aby nemuselo docházet k jejich replikaci. Hierarchie je pak nastavena tak, aby informace uložené v jednom portálu byly v maximálním možném počtů případů pro druhý portál dostupné pouze pro čtení. Jako příklad je možné uvést informace o spolupracujících organizacích nebo jednotlivých výzvách pro podávání žádostí, které jsou důležité jak pro podávání žádostí, tak pro existující projekty.

6. Nástroje použité při vývoji

V následující kapitole bychom rádi zmínili nejdůležitější nástroje použité při vývoji nového informačního systému.

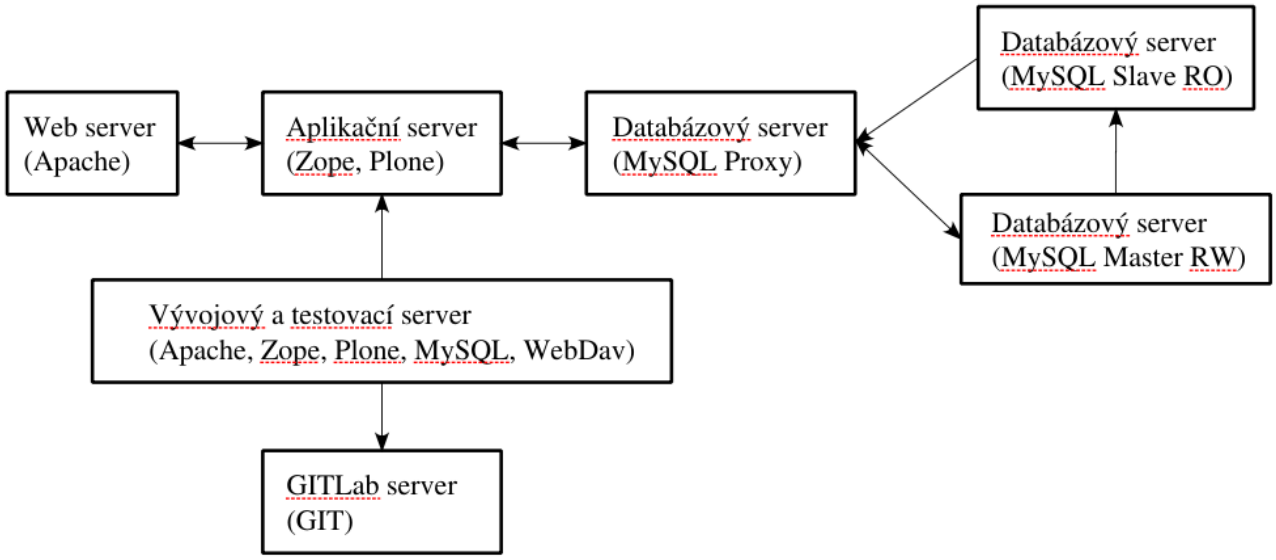
Některé z nich, jako například systém správy verzí, jsou při vývoji obsáhlých informačních systémů standardem. Jiné však nejsou u vývoje informačních systémů běžně používány.

6.1 Serverová infrastruktura

Následující obrázek ilustruje logické schéma použité serverové infrastruktury včetně části použitých komponent a samostatné instance nového informačního systému použité pro vývoj a testování

Obrázek 3 Serverová infrastruktura

Šipky ve schématu označují toky dat mezi jednotlivými komponentami informačního systému.



6.2 Aplikační server Zope

Zope [3] je převážně v jazyce Python naprogramovaný *opensource* aplikační server, který ve své transakční objektové databázi obsahuje mimo vlastního webového obsahu také skripty psané také v jazyce Python, dynamické šablony pro generování HTML výstupu a další obsah. Zope přímo podporuje hojně využívané napojení na relační databázové systémy.

Administrace aplikačního serveru Zope probíhá skrze webové rozhraní, které umožňuje také úpravy obsahu ve vestavěném textovém editoru, který podporuje zamykání souboru a tím serializaci přístupu jednotlivých administrátorů.

Aplikační server Zope lze snadno rozšířit množstvím tzv. produktů, které základnímu aplikačnímu serveru zpřístupní různé funkcionality na základě individuálních potřeb. Mezi nejznámější patří konektory na jednotlivé databáze a úložiště nebo například námi využitý produkt na správu obsahu Plone.

Vzhledem k potřebě vytvoření a sdílení společných funkcionalit mezi jednotlivých portály ekosystému superpočítačového centra jsme v rámci vývoje nového informačního systému vytvořili také vlastní produkt pro popsaný aplikační server. Jeho strukturou a obsahem se budu zabývat v následujících kapitolách.

6.2.1 Šablonovací jazyk

Aplikační server Zope používá pro dynamické vytváření výstupů v HTML či XML systém TAL - tedy Template Atribute Language. Tento jazyk umožňuje pomocí konstrukcí psaných v jazyce XML dynamicky generovat výstupní soubory z připravených šablon, přičemž je možné jednotlivé elementy výstupu měnit, vytvářet nové, opakovat nebo je třeba úplně vynechat.

Mezi nejčastější používané konstrukce patří:

- *tal:attributes* - dynamicky změní atribut daného elementu,
- *tal:define* - nadefinuje novou proměnnou,
- *tal:condition* - otestuje zadanou podmínku,
- *tal:content* - zamění obsah elementu,
- *tal:on-error* - zachytí chybu,
- *tal:repeat* - zopakuje jeden element vícekrát.

TAL je velice silný nástroj a s jeho pomocí lze vytvářet i velmi komplexní šablony. Je ale třeba mít na paměti, že by šablony neměly být příliš složité a neměla by se do nich přesouvat logika vhodná spíše pro aplikační vrstvu informačního systému.

6.2.2 Skripty

Skripty jsou v aplikačním serveru Zope, stejně jako aplikační server samotný, psány v jazyce Python. Použití v rámci tohoto aplikačního serveru sebou ovšem nese jistá omezení v podobě omezené funkcionality a nemožnosti použít některé standardní knihovny. Toto omezení je možné obejít vytvořením vlastního balíčku, ze kterého je pak možné importovat do skriptu třídy a funkce.

Skriptům se stejně jako dalším objektům v objektově orientovaném aplikačním serveru Zope dají nastavit přístupová oprávnění na jednotlivé role a tím efektivně omezit skupiny uživatelů, kteří budou moci skripty spouštět.

Nezbytnou a velmi často používanou funkcionalitou je možnost manipulace s objekty reprezentujícími požadavek přicházející od uživatele (*request*) a odpověď odesílaná uživateli (response). Je tak možné ze skriptu například nabídnout uživateli stažení souboru změnou *content-type* v odpovědi uživateli.

Dalšími velmi často užívanými možnostmi jsou manipulace se sezením uživatele či spouštění externích skriptů či SQL metod.

6.2.3 SQL metody

SQL metody jsou samostatnou částí aplikačního serveru Zope a ve spojení s předem definovaným napojením na relační databázový systém je možné s jejich pomocí pracovat s nejrůznějšími databázemi.

Jak již bylo zmíněno v kapitole o možných útocích na webové aplikace, používá se k tvorbě dynamických SQL metod šablonovací jazyk DTML, který umožňuje bezpečné vkládání parametrů získaných od uživatele přímo do SQL metody.

SQL metoda jako taková pak může kromě logiky samotného jazyka SQL obsahovat také logiku podporovanou jazykem DTML.

6.3 Plone

Plone [4] je *opensource* systém určený primárně pro správu obsahu a k tvorbě intranetových a extranetových portálů běžnými uživateli. Jednou z jeho největších předností je množství doplňků, kterými se dá Plone přizpůsobit pro různé případy užití a napojit na celou řadu externích systémů.

V případě nového informačního systému jsme se rozhodli vybrat Plone jako výchozí bod, který nám umožní využít jeho již existující funkce, mezi které patří například řešení oprávnění, napojení na LDAP, správa obsahu apod., a zároveň máme stále možnost přístupu k nízkoúrovňovému programování důležitých částí informačního systému.

Plone je stejně jako aplikační server Zope, jehož objektovou architekturu Plone využívá, naprogramován v jazyce Python a stejný jazyk je tedy využit pro tvorbu celého informačního systému.

6.4 DavFS

WebDAV (Web-based Distributed Authoring and Versioning) je množina rozšíření HTTP protokolu, která umožňuje spravovat soubory na webovém serveru skrze protokol HTTP.

Kromě vytváření, kopírování, mazání a přesouvání je zde implementován i důležitý mechanismus uzamykání souborů, který řeší serializaci přístupu k souborům a zabraňuje tak konfliktům vzniklým při editaci jednoho souboru více vývojáři.

V našem případě využíváme WebDAV, resp. jeho linuxovou implementaci davfs2 z toho důvodu, že aplikační server Zope, na kterém je nový informační systém postaven, spoléhá při správě souborů na vlastní interní databázi, kde jsou všechny soubory uloženy ve formě objektů a lze k nim přistupovat jen skrze ZMI (Zope Management Interface). Toto řešení je sice univerzální a umožňuje snadnou editaci souborů i ve webovém prohlížeči, ale neumožňuje pracovat se soubory stejným způsobem jako by byly uloženy na souborovém systému. To způsobuje potíže především při potřebě využití pokročilejších editorů nebo při potřebě správy verzí obsahu mimo databázi aplikačního serveru.

6.5 GIT + GITLab

Systém správy verzí umožňující ukládání jednotlivých změn v kódu aplikace a v kteroukoli chvíli návrat ke kterékoli předchozí verzi je při vývoji jakéhokoli systému nezbytným pomocníkem. Při vývoji nového informačního systému jsme se rozhodli pro použití nejznámějšího a nejrozšířenějšího verzovacího

systému GIT, který stejně jako Linuxové jádro vyvinul Linus Torvalds.

GIT je nízkoúrovňový systém správy verzí, který je mezi verzovacími systémy nejrozšířenější pro svou rychlost, škálovatelnost, decentralizovanost a široké možnosti použití.

V decentralizovaném vývoji pomocí verzovacího systému je často používán centrální server pro ukládání výsledků vývoje jednotlivých vývojářů a správu projektu jako celku. Pro tyto případy je v komunitě okolo otevřeného software nejčastěji používán portál GitHub. Pro splnění stejných potřeb v privátním sektoru se dá využít *opensource* projekt naprogramovaný v jazyce Ruby jménem GitLab [5]. GitLab poskytuje webové rozhraní pro vizualizaci záznamů z verzovacího systému, správu přístupových oprávnění k jednotlivým projektům a také možnosti řešení pro dokumentaci jednotlivých projektů a pro řešení požadavků uživatelů k jednotlivým projektům.

6.6 Apache

Webový server Apache [6] slouží v tomto prostředí nejen jako běžný webový server zprostředkovávající požadavky uživatele aplikačnímu serveru a odpovědi zpět z aplikačního serveru uživateli, ale stará se také o dvě důležité části využívané hlavně při registraci do portálu - zpracování uživatelského SSL certifikátu, resp. zpracování přihlášení uživatele skrze eduID.

Webový server je také oddělen od aplikačního serveru a tím zajišťuje případnou možnost škálování celého řešení napříč více webovými a eventuálně i více aplikačními servery.

6.6.1 SSL certifikáty

Pokud si uživatel vybere možnost registrace uživatelského účtu pomocí SSL certifikátu, je odkázán na speciální adresu, která je právě tímto webovým serverem odchycena a je pro ni použito odlišné nastavení.

Takto nastavený webový server provede na daném umístění validaci uživatelského certifikátu až deset úrovní nahoru v řetězu certifikačních autorit a následně kromě informace o validitě daného certifikátu předá i důležité informace v něm obsažené aplikačnímu serveru, který je využije pro předvyplnění registračního formuláře, resp. zobrazení chyby, pokud je uživatelský certifikát neplatný. Abychom se z hlediska bezpečnosti vyhnuli možnému riziku vyplývajícímu z ruční editace formulářových polí případným útočníkem, je uživatelský certifikát ověřován ještě jednou v okamžiku odeslání formuláře.

Portál jako celek je samozřejmě provozován na HTTPS s důvěryhodným certifikátem, což zajišťuje bezpečnost komunikace mezi uživatelem a serverem.

6.6.2 Shibboleth

Shibboleth je celosvětově nejpoužívanější open source projekt využívaný pro jednotné přihlašování uživatelů k více síťovým zdrojům. V našem případě je Shibboleth využívaný českou federací identit eduID, skrze kterou umožňujeme uživatelům z jiných českých akademických institucí registraci do nového informačního systému.

Shibboleth úzce spolupracuje právě s webovým serverem, který mu předá informace o nutnosti přihlášení uživatele a po jeho provedení jsou informace o korektním přihlášení a další vyžadovaná data předána zpět webovému serveru.

Samotné přihlášení pak funguje na straně poskytovatele identit, kam je uživatel přesměrován a kde dochází k faktickému ověření jeho účtu. Po úspěšném přihlášení od poskytovatele identit přenese k poskytovateli služby informace o úspěšném přihlášení a uživatel je přesměrován zpět. Takto vytvořenému sezení je pak možno dle parametrů získaných od poskytovatele identit přizpůsobit roli daného uživatele.

6.6.3 Jednotné přihlášení

Protože se při implementaci malého a v současné chvíli i velkého superpočítače neustále rozrůstá množina podpůrných služeb a nástrojů, bylo žádoucí společně s nově vznikajícím informačním systémem implementovat také možnost, jak mezi tímto a dalšími systémy přecházet bez nutnosti nového přihlášení na každý portál zvlášť.

V rámci implementace nového informačního systému jsme tedy vytvořili také systém jednotného přihlášení, který využívá funkcionality webového serveru Apache a v něm zakomponovaný modul *mod_auth_tkt*. Tento modul je jednoduchý a umožňuje implementaci jednotného přihlášení použitím bezpečných lístků uložených v *cookies* prohlížeče. Modul mimo jiné podporuje časové omezení platnosti *cookie*, možnost různého pojmenování *cookie* a také uložení uživatelských dat do *cookie* samotné.

Celý proces ověření funguje tak, že webový server přijme od uživatele v rámci požadavku na daný portál jeho autentizační token obsažený v *cookie*. Tento token pak porovná s MD5 kontrolním součtem generovaným z uživatelského jména, dalších dat o uživateli a sdíleného tajemství serveru. Pokud se kontrolní součty shodují, je uživateli zobrazen požadovaný obsah. V opačném případě je uživatel přesměrován na přihlašovací stránku jednoho z portálů, kde se mu po přihlášení tato autentizační *cookie* vygeneruje.

Konfigurace pro sekci portálu se statistikami pak může vypadat například takto:

```
    AuthType Basic
    require valid-user
    TKTAuthBackCookieName TKTAuthBackCookie
    TKTAuthLoginURL
https://extranet.it4i.cz/acl_users/credentials_cookie_auth/require_login?came_from=http%3A//extranet.it4i.cz/piwik
    TKTAuthCookieName __ac
    TKTAuthTimeout 0
    TKTAuthIgnoreIP on
```

6.6.4 Balancing

I přes použití jednoduché serverové infrastruktury a rozdělení jednotlivých komponent na separátní servery je možné provádět balancování provozu mezi webovým a aplikačním serverem, díky čemuž bude opět o něco zvýšena dostupnost takového řešení a tato implementace pak bude připravena pro další případné rozšíření.

V našem případě máme pro webový server a aplikační server vyhrazen vždy jeden fyzický server. Aby se nemohlo stát, že bude instance aplikačního serveru zahlcena požadavky jednoho nebo více klientů do té míry, že by se na požadavky ostatních nedostalo, využijeme možností balancování provozu

webového serveru Apache pomocí interního Proxy Balanceru a na straně aplikačního serveru spustíme dvě instance, z nichž každá bude naslouchat na jiném TCP portu.

Základem konfigurace je následující direktiva:

```
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/"
env=BALANCER_ROUTE_CHANGED
<Proxy balancer://httpsplone>
   BalancerMember http://plone.it4i.cz:8080 route=1
   BalancerMember http://plone.it4i.cz:8081 route=2
   ProxySet stickysession=ROUTEID
</Proxy>
```

Tato část konfigurace nám nejdříve nastavuje cookie pro každého klienta tak, aby se při prvním přístupu na jednu z instancí aplikačního serveru tato informace uložila do ROUTEID a každý další požadavek se dostal ke stejné instanci. Toto je důležité především proto, že po přihlášení je uživatelovo sezení udržováno pouze tou instancí, ke které se přihlásil, a v případě změny instance by došlo k odhlášení sezení.

Následující konfigurační blok nám do Balanceru registruje jeho členy a tedy dvě instance téhož aplikačního serveru, ovšem jen s rozdílnými porty.

Následně se pak pro libovolnou podmínku (RewriteCond) použije následující pravidlo (RewriteRule), které provoz přesměruje právě skrze Ballancer:

```
RewriteRule ^/(.*)
balancer://httpsplone/VirtualHostBase/https/extranet.it4i.cz:443/extranet/VirtualHostRoot/$1 [L,P]
```

6.7 MySQL

Jako databázová vrstva pro nově vznikající informační systém byla zvolena databáze MySQL [7] především kvůli předchozím pozitivním zkušenostem a také existujícímu použití na jiných portálech v ekosystému SCC.

V následujících dvou kapitolách si popíšeme nastavení a používání MySQL databázové vrstvy pro informační systém.

6.7.1 MySQL Proxy

Základním stavebním kamenem a také samostatným serverem je v našem případě MySQL Proxy. MySQL Proxy je samostatný softwarový produkt vyvinutý pro zprostředkování spojení mezi jedním nebo více klienty a jedním nebo více servery. V tomto případě má MySQL Proxy k dispozici dva samostatné databázové servery a přistupuje na ni několik samostatných klientů.

Toto řešení je v současném stavu bohatě dostačující a k jeho realizaci vedla potřeba mít robustní databázové řešení s možností budoucího škálování stejně jako v případě webového serveru.

MySQL Proxy je nakonfigurována tak, aby jednu z databází využívala pro čtení i zápis, zatímco

druhou pouze pro čtení. Tato konfigurace je z důvodu replikace mezi databázemi popsána dále.

6.7.2 MySQL replikace

Mezi dvěma výše zmíněnými databázovými servery probíhá online replikace implementována přímo v MySQL.

Celý proces funguje tak, že je možné se obou databází dotazovat a číst z nich data, ale pouze do jedné z nich je možné zapisovat. Takto provedené změny jsou MySQL serverem ukládány do binárního logu a jsou v reálném čase přenášeny na druhý databázový server a zde jsou aplikovány na druhou databázi.

Kdyby byl povolen zápis i do druhé databáze, mohlo by při aplikaci změn dojít k nekonzistenci stavů databází a replikace by přestala fungovat. To by v krajním případě znamenalo, že by uživatel při dvou stejných dotazech dostal dva různé výsledky.

V tomto případě je ale takové řešení replikace s použitím proxy naprosto dostačující, protože majoritní množství provozu zpracovaného databázovou vrstvou je právě čtení dat a zápis tak při současné zátěži bez potíží zvládne zpracovat jen jeden ze dvou serverů.

7. Zdrojové kódy

V této části se konkrétněji zaměříme na zdrojové kódy aplikační, prezentační a databázové vrstvy a popíšeme si jejich nejdůležitější vlastnosti.

7.1 Struktura databází

V následujících dvou kapitolách se zaměříme na popis struktury dvou nejúžeji spolupracujících databází a tedy na databázi nově vznikajícího informačního systému a databázi existujícího portálu SCCS.

7.1.1 SCCS portál

Tato kapitola obsahuje současnou strukturu databáze navazujícího portálu SCCS, která ilustruje množství informací v ní uložených a sdílených mezi jednotlivými systémy.

Tabulky obsažené v této databázi mají následující účel:

- *accounting_** - tyto tabulky jsou použity pro záznam úloh, uživatelů a využitých jádrohodin jako výstup z plánovače superpočítače;
- *call* - obsahuje informace o jednotlivých výzvách a jejich časovém rozvrhu;
- *call_type* - obsahuje informace o typech výzev a je úzce spjata s předchozí tabulkou;
- *events* - slouží k logování událostí vyvolaných uživatelem v portálu;
- *infrastructure_power_consumption* - slouží k ukládání informací o spotřebě elektrické energie clusteru;
- *organization* - obsahuje seznam organizací pro spárování s projekty;
- *overview_utilization* - obsahuje celkové využití clusteru;
- *pbs_exit_codes* - obsahuje číselník návratových hodnot plánovače;
- *project* - tabulka obsahuje informace o všech projektech;
- *project_area* - tabulka je provázaná s výše zmíněnou a obsahuje informace o oblasti výzkumu daného projektu;
- *project_budgets* - tabulka určená pro accounting projektů PRACE;
- *project_prace* - tabulka obsahující identifikátory projektů přicházejících do našeho centra ze sítě PRACE;
- *queue* - obsahuje informace o frontách použitých v plánovači superpočítače;
- *queue_project* - vazební tabulka mezi projekty a frontami, které smí projekt využít;
- *queue_project_oor* - stejný případ jako výše s tím rozdílem, že se jedná o fronty, které může projekt využít po vyčerpání celé své alokace;
- *reservation_project* - vazební tabulka spojující informace o statických alokacích vzhledem

k projektu;

- *setting* - tabulka obsahující nastavení, především verzi databáze;
- *usagerecord* - tabulka určená pro accounting projektů PRACE;
- *user_project* - vazební tabulka přiřazující uživatele k jednotlivým projektům.

```
CREATE TABLE `accounting_login` (
  `date` date NOT NULL,
  `login` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `time_spent` time NOT NULL,
  UNIQUE KEY `date` (`date`,`login`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `accounting_pbs` (
  `pid` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
  `login` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
  `core_hours` double NOT NULL DEFAULT '0',
  UNIQUE KEY `pid` (`pid`(62),`login`(62)),
  KEY `pid_2` (`pid`),
  KEY `login` (`login`),
  KEY `core_hours` (`core_hours`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `accounting_pbs_job` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `account` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `ctime` int(11) DEFAULT NULL,
  `end` int(11) DEFAULT NULL,
  `etime` int(11) DEFAULT NULL,
  `exec_host` text COLLATE utf8_czech_ci,
  `exec_vnode` text COLLATE utf8_czech_ci,
  `exit_status` int(11) DEFAULT NULL,
  `group` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `jobdatetime` datetime NOT NULL,
  `jobid` varchar(63) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  `jobname` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `jobstate` varchar(63) COLLATE utf8_czech_ci NOT NULL,
  `project` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `qtime` int(11) DEFAULT NULL,
  `queue` varchar(63) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `resvid` varchar(63) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `resvname` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `rl_host` varchar(63) COLLATE utf8_czech_ci DEFAULT NULL,
  `rl_mem` varchar(63) COLLATE utf8_czech_ci DEFAULT NULL,
  `rl_mpiprocs` int(11) DEFAULT NULL,
  `rl_ncpus` int(11) DEFAULT NULL,
  `rl_nodect` int(11) DEFAULT NULL,
  `rl_nodes` varchar(255) COLLATE utf8_czech_ci DEFAULT NULL,
```

```
  `rl_place` varchar(63) COLLATE utf8_czech_ci DEFAULT NULL,
  `rl_pmem` varchar(63) COLLATE utf8_czech_ci DEFAULT NULL,
  `rl_pvmem` varchar(63) COLLATE utf8_czech_ci DEFAULT NULL,
  `rl_qlist` varchar(63) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `rl_select` text COLLATE utf8_czech_ci,
  `rl_vmem` varchar(63) COLLATE utf8_czech_ci DEFAULT NULL,
  `rl_walltime` time DEFAULT NULL,
  `ru_cpupercent` int(11) DEFAULT NULL,
  `ru_cput` time DEFAULT NULL,
  `ru_mem` varchar(63) COLLATE utf8_czech_ci DEFAULT NULL,
  `ru_ncpus` int(11) DEFAULT NULL,
  `ru_vmem` varchar(63) COLLATE utf8_czech_ci DEFAULT NULL,
  `ru_walltime` time DEFAULT NULL,
  `run_count` int(11) DEFAULT NULL,
  `session` int(11) DEFAULT NULL,
  `start` int(11) DEFAULT NULL,
  `user` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `jobid` (`jobid`,`run_count`),
  KEY `exit_status` (`exit_status`),
  KEY `jobdatetime` (`jobdatetime`),
  FULLTEXT KEY `jobid_2` (`jobid`,`account`,`group`,`jobname`,`queue`,`user`)
) ENGINE=InnoDB AUTO_INCREMENT=524816 DEFAULT CHARSET=utf8
COLLATE=utf8_czech_ci;

DELIMITER ;;
/*!50003 CREATE*/ /*!50017 DEFINER=`root`@`localhost`*/ /*!50003 TRIGGER
`accounting_pbs_job_after_insert` AFTER INSERT ON `accounting_pbs_job` FOR EACH ROW
BEGIN
  IF NEW.queue IN ( SELECT name FROM queue WHERE projectless = 0 ) THEN
    INSERT INTO accounting_pbs (`pid`, `login`, `core_hours`) VALUES (IFNULL(NEW.account, ''),
IFNULL(NEW.user, ''), (TIME_TO_SEC(IFNULL(NEW.ru_walltime, 0)) * IFNULL(NEW.rl_ncpus, 0))
/ 3600) ON DUPLICATE KEY UPDATE core_hours = core_hours +
(TIME_TO_SEC(IFNULL(NEW.ru_walltime, 0)) * IFNULL(NEW.rl_ncpus, 0)) / 3600;
    IF NEW.account IN ( SELECT pid FROM project_prace ) THEN
      INSERT IGNORE INTO usagerecord VALUES ( CONCAT('VSB-TUO-
',SHA1(CONCAT(NEW.user,NEW.jobid,CONVERT_TZ(NEW.jobdatetime,'CET','UTC')))),( SELECT
name FROM project WHERE pid = NEW.account
),NEW.user,NEW.user,CONVERT_TZ(NEW.jobdatetime,'CET','UTC'),NEW.jobid,NEW.jobname,CAS
E WHEN NEW.jobstate = 'E' THEN 'completed' WHEN NEW.jobstate = 'R' THEN 'started' ELSE
NEW.jobstate
END,CONVERT_TZ(FROM_UNIXTIME(NEW.ctime),'CET','UTC'),CONVERT_TZ(FROM_UNIXTI
ME(NEW.start),'CET','UTC'),CONVERT_TZ(FROM_UNIXTIME(NEW.end),'CET','UTC'),'anselm.it4i.
cz','Anselm',SUBSTRING_INDEX(NEW.exec_host,'/',1),NEW.rl_ncpus,'1.0',NEW.ru_ncpus,TIME_TO
_SEC(NEW.ru_walltime),TIME_TO_SEC(NEW.ru_cput),'PRACE' );
    END IF;
  END IF;
END */;;
DELIMITER ;
```

```
CREATE TABLE `accounting_pbs_reservation` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `auth_groups` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `auth_users` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `ctime` int(11) DEFAULT NULL,
  `duration` int(11) DEFAULT NULL,
  `end` int(11) DEFAULT NULL,
  `nodes` text COLLATE utf8_czech_ci,
  `owner` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `queue` varchar(63) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `resvdatetime` datetime NOT NULL,
  `resvid` varchar(63) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  `resvname` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `resvstate` varchar(63) COLLATE utf8_czech_ci NOT NULL,
  `rl_host` varchar(63) COLLATE utf8_czech_ci DEFAULT NULL,
  `rl_mem` varchar(63) COLLATE utf8_czech_ci DEFAULT NULL,
  `rl_mpiprocs` int(11) DEFAULT NULL,
  `rl_ncpus` int(11) DEFAULT NULL,
  `rl_nodect` int(11) DEFAULT NULL,
  `rl_place` varchar(63) COLLATE utf8_czech_ci DEFAULT NULL,
  `rl_select` text CHARACTER SET utf8 COLLATE utf8_bin,
  `rl_vmem` varchar(63) COLLATE utf8_czech_ci DEFAULT NULL,
  `rl_walltime` time DEFAULT NULL,
  `start` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `resvid` (`resvid`),
  KEY `resvdatetime` (`resvdatetime`),
  FULLTEXT KEY `jobid_2` (`resvid`,`resvname`,`auth_groups`,`queue`,`auth_users`)
) ENGINE=InnoDB AUTO_INCREMENT=350 DEFAULT CHARSET=utf8
COLLATE=utf8_czech_ci;

CREATE TABLE `call` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `call_number` int(11) NOT NULL,
  `type_id` int(11) NOT NULL,
  `submission_start` datetime NOT NULL,
  `submission_end` datetime NOT NULL,
  `announcement` datetime NOT NULL,
  `runtime_start` datetime NOT NULL,
  `runtime_end` datetime NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=17 DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `call_type` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `prefix` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

```
CREATE TABLE `events` (
  `time` datetime DEFAULT CURRENT_TIMESTAMP,
  `section` varchar(255) COLLATE utf8_czech_ci DEFAULT NULL,
  `username` varchar(255) COLLATE utf8_czech_ci DEFAULT NULL,
  `message` text COLLATE utf8_czech_ci
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `infrastructure_power_consumption` (
  `date_hour` datetime NOT NULL,
  `main_measure` double NOT NULL,
  `ict_small_cluster` double NOT NULL,
  `specialized_service` double NOT NULL,
  `specialized_mobull` double NOT NULL,
  PRIMARY KEY (`date_hour`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `organization` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `identification_number` varchar(20) COLLATE utf8_czech_ci DEFAULT NULL,
  `hidden` tinyint(1) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `organization_name` (`name`),
  FULLTEXT KEY `name` (`name`)
) ENGINE=InnoDB AUTO_INCREMENT=165 DEFAULT CHARSET=utf8
COLLATE=utf8_czech_ci;

CREATE TABLE `overview_utilization` (
  `date` date NOT NULL,
  `cores_allocated` decimal(10,0) NOT NULL,
  `cores_utilized` decimal(10,0) NOT NULL,
  PRIMARY KEY (`date`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `pbs_exit_codes` (
  `exit_code` int(11) NOT NULL,
  `name` varchar(63) COLLATE utf8_czech_ci NOT NULL,
  `reason` varchar(63) COLLATE utf8_czech_ci NOT NULL,
  `description` text COLLATE utf8_czech_ci NOT NULL,
  PRIMARY KEY (`exit_code`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `project` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` tinytext COLLATE utf8_czech_ci NOT NULL,
  `pid` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
  `start_time` datetime NOT NULL,
  `end_time` datetime NOT NULL,
  `type_id` int(11) NOT NULL,
```

```
  `affiliation_id` int(11) NOT NULL,
  `area_id` int(11) NOT NULL,
  `pi` tinytext COLLATE utf8_czech_ci NOT NULL,
  `pi_login` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
  `pi_salutation` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `pi_mail` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `resources` int(11) NOT NULL DEFAULT '0',
  `total_score` int(11) NOT NULL DEFAULT '0',
  `allocation` int(11) NOT NULL DEFAULT '0',
  `call` int(11) NOT NULL DEFAULT '0',
  `active` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`),
  UNIQUE KEY `pid` (`pid`),
  KEY `type_id` (`type_id`),
  KEY `area_id` (`area_id`),
  FULLTEXT KEY `name` (`name`,`pi`,`pi_mail`)
) ENGINE=InnoDB AUTO_INCREMENT=479 DEFAULT CHARSET=utf8
COLLATE=utf8_czech_ci;

DELIMITER ;;
/*!50003 CREATE*/ /*!50017 DEFINER=`root`@`localhost`*/ /*!50003 TRIGGER
`project_before_insert` BEFORE INSERT ON `project` FOR EACH ROW SET @prefix =
CONCAT_WS('-',(SELECT prefix FROM call_type WHERE id = NEW.type_id),IF(NEW.type_id <> 4,
NEW.`call`, DATE_FORMAT(NOW(),"%y"))),
NEW.pid = IF(NEW.type_id <> 0,
            CONCAT_WS('-',
                  @prefix,
                  IFNULL((SELECT MAX(CONVERT(SUBSTRING_INDEX(pid,'-',-1),UNSIGNED
INTEGER))+1 FROM project WHERE pid LIKE CONCAT(@prefix,'%')),1)
                 ),
            REPLACE(TRIM(UPPER(NEW.name)), ' ', '-')
           ),
NEW.start_time = IF(NEW.start_time='0000-00-00 00:00:00', NOW(), NEW.start_time),
NEW.end_time = IF(NEW.end_time='0000-00-00 00:00:00', ADDDATE(NOW(), INTERVAL 1
MONTH), NEW.end_time) */;;
DELIMITER ;

CREATE TABLE `project_area` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`)
) ENGINE=InnoDB AUTO_INCREMENT=30 DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `project_budgets` (
  `id` int(11) NOT NULL,
  `name` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `site` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `machine` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `budget` bigint(20) DEFAULT NULL,
```

```
  `project_start` date DEFAULT NULL,
  `project_end` date DEFAULT NULL,
  PRIMARY KEY (`id`,`site`(166),`machine`(166))
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `project_prace` (
  `pid` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  PRIMARY KEY (`pid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `queue` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  `projectless` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`)
) ENGINE=InnoDB AUTO_INCREMENT=147 DEFAULT CHARSET=utf8
COLLATE=utf8_czech_ci;

CREATE TABLE `queue_project` (
  `queue_id` int(11) NOT NULL,
  `project_id` int(11) NOT NULL,
  UNIQUE KEY `queue_id` (`queue_id`,`project_id`),
  KEY `queue_id_2` (`queue_id`),
  KEY `project_id` (`project_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `queue_project_oor` (
  `queue_id` int(11) NOT NULL,
  `project_id` int(11) NOT NULL,
  UNIQUE KEY `queue_id` (`queue_id`,`project_id`),
  KEY `queue_id_2` (`queue_id`),
  KEY `project_id` (`project_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `reservation_project` (
  `reservation_id` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `project_id` int(11) NOT NULL,
  UNIQUE KEY `reservation_id` (`reservation_id`,`project_id`),
  KEY `reservation_id_2` (`reservation_id`),
  KEY `project_id` (`project_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `setting` (
  `key` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `value` text COLLATE utf8_czech_ci NOT NULL,
  PRIMARY KEY (`key`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `usagerecord` (
```

```
  `recordid` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `projectname` varchar(255) COLLATE utf8_czech_ci DEFAULT NULL,
  `globaluserid` varchar(255) COLLATE utf8_czech_ci DEFAULT NULL,
  `localuserid` varchar(255) COLLATE utf8_czech_ci DEFAULT NULL,
  `createtime` datetime DEFAULT NULL,
  `localjobid` varchar(255) COLLATE utf8_czech_ci DEFAULT NULL,
  `jobname` varchar(255) COLLATE utf8_czech_ci DEFAULT NULL,
  `status` varchar(32) COLLATE utf8_czech_ci DEFAULT NULL,
  `submittime` datetime DEFAULT NULL,
  `starttime` datetime DEFAULT NULL,
  `endtime` datetime DEFAULT NULL,
  `submithost` varchar(255) COLLATE utf8_czech_ci DEFAULT NULL,
  `machinename` varchar(255) COLLATE utf8_czech_ci DEFAULT NULL,
  `host` varchar(255) COLLATE utf8_czech_ci DEFAULT NULL,
  `processors` int(11) DEFAULT NULL,
  `consumptionrate` double DEFAULT NULL,
  `usedprocessors` int(11) DEFAULT NULL,
  `wallduration` bigint(20) DEFAULT NULL,
  `cpuduration` bigint(20) DEFAULT NULL,
  `domain` varchar(255) COLLATE utf8_czech_ci DEFAULT NULL,
  PRIMARY KEY (`recordid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `user_project` (
  `login` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  `project_id` int(11) NOT NULL,
  UNIQUE KEY `pair` (`login`,`project_id`),
  KEY `login` (`login`),
  KEY `project_id` (`project_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

7.1.2 Extranet portál

Vzhledem k úzkému propojení mezi jednotlivými portály jsou i data částečně sdílena a tak není potřeba jejich duplikace mezi jednotlivými databázemi. Databáze nově vznikajícího portálu tak obsahuje převážně data, která ještě nejsou uložena v jiné navázané databázi.

Konkrétně se jedná o následující tabulky a jejich účel:

- *client_cert* - tabulka obsahuje otisky klientských certifikátů použitých při registraci, které se používají při detekci pokusu o opětovnou registraci s využitím stejného certifikátu;
- *client_eppn* - stejný případ jako u tabulky výše, jen s tím rozdílem, že zde se ukládají otisky informací získaných z eduID;
- *evaluator_committee* - tabulka obsahující informace o hodnotitelích a druzích výzev, ke kterým jsou jako hodnotitelé přiřazeni;
- *events* - tabulka sloužící k zaznamenání událostí prováděných uživateli v portálu;
- *project_queue* - tato tabulka obsahuje projektové žádosti připravené ke zpracování v portále SCCS, kde se ze žádostí vytváří projekt;

- *project_requests* - tabulka obsahující aktuálně podané a zpracovávané projektové žádosti;
- *project_requests_archive* - tabulka obsahuje projektové žádosti, které již byly zpracovány;
- *request_attachment* - samostatná tabulka pro přílohy projektových žádostí;
- *request_comments* - do této tabulky se ukládají jednotlivé komentáře hodnotitelů k projektovým žádostem;
- *request_commissioner* - vazební tabulka pro přiřazení hodnotitele ke konkrétní žádosti k jejímu hodnocení;
- *setting* - tabulka obsahující dodatečná nastavení; aktuálně obsahuje především současnou verzi databáze umožňující její verzování a aplikaci aktualizací databázové struktury.

Následuje kompletní SQL kód popisující strukturu zmíněné databáze.

```
CREATE TABLE `client_cert` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(255) COLLATE utf8_bin NOT NULL,
  `sha1sum` varchar(255) COLLATE utf8_bin NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `sha1sum` (`sha1sum`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

CREATE TABLE `client_eppn` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `eppn` varchar(255) COLLATE utf8_bin NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `eppn` (`eppn`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

CREATE TABLE `evaluator_committee` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `login` varchar(255) COLLATE utf8_bin NOT NULL,
  `internal` tinyint(4) NOT NULL DEFAULT '0',
  `open` tinyint(4) NOT NULL DEFAULT '0',
  `dd` tinyint(4) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`),
  UNIQUE KEY `login` (`login`)
) ENGINE=InnoDB AUTO_INCREMENT=53 DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

CREATE TABLE `events` (
  `time` datetime DEFAULT NULL,
  `section` varchar(255) COLLATE utf8_czech_ci DEFAULT NULL,
  `username` varchar(255) COLLATE utf8_czech_ci DEFAULT NULL,
  `message` text COLLATE utf8_czech_ci,
  KEY `time` (`time`),
  KEY `section` (`section`),
  KEY `username` (`username`),
  KEY `message` (`message`(255))
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `project_queue` (
  `id` int(11) NOT NULL,
  `name` tinytext COLLATE utf8_czech_ci NOT NULL,
  `call_id` int(11) NOT NULL,
  `resources` int(11) NOT NULL,
  `allocation` int(11) NOT NULL DEFAULT '0',
  `additional_resources` tinytext COLLATE utf8_czech_ci,
  `research_area_id` int(11) NOT NULL,
  `pi_login` tinytext COLLATE utf8_czech_ci NOT NULL,
  `pi_name` tinytext COLLATE utf8_czech_ci NOT NULL,
  `pi_salutation` tinytext COLLATE utf8_czech_ci NOT NULL,
  `institute` int(11) NOT NULL,
  `address` tinytext COLLATE utf8_czech_ci,
  `popular_abstract` text COLLATE utf8_czech_ci NOT NULL,
  `attachment_name` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `attachment_type` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `created` datetime NOT NULL,
  `last_change` datetime NOT NULL,
  `deadline` datetime NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `project_requests` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` tinytext COLLATE utf8_czech_ci NOT NULL,
  `call_id` int(11) NOT NULL,
  `resources` int(11) NOT NULL,
  `allocation` int(11) NOT NULL DEFAULT '0',
  `additional_resources` tinytext COLLATE utf8_czech_ci,
  `research_area_id` int(11) NOT NULL,
  `pi_login` tinytext COLLATE utf8_czech_ci NOT NULL,
  `pi_name` tinytext COLLATE utf8_czech_ci NOT NULL,
  `pi_salutation` tinytext COLLATE utf8_czech_ci NOT NULL,
  `institute` int(11) NOT NULL,
  `address` tinytext COLLATE utf8_czech_ci,
  `popular_abstract` text COLLATE utf8_czech_ci NOT NULL,
  `attachment_name` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `attachment_type` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `created` datetime NOT NULL,
  `last_change` datetime NOT NULL,
  `deadline` datetime NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=101 DEFAULT CHARSET=utf8
COLLATE=utf8_czech_ci;

CREATE TABLE `project_requests_archive` (
  `id` int(11) NOT NULL,
  `name` tinytext COLLATE utf8_czech_ci NOT NULL,
```

```
  `call_id` int(11) NOT NULL,
  `resources` int(11) NOT NULL,
  `allocation` int(11) NOT NULL DEFAULT '0',
  `additional_resources` tinytext COLLATE utf8_czech_ci,
  `research_area_id` int(11) NOT NULL,
  `pi_login` tinytext COLLATE utf8_czech_ci NOT NULL,
  `pi_name` tinytext COLLATE utf8_czech_ci NOT NULL,
  `pi_salutation` tinytext COLLATE utf8_czech_ci NOT NULL,
  `institute` int(11) NOT NULL,
  `address` tinytext COLLATE utf8_czech_ci,
  `popular_abstract` text COLLATE utf8_czech_ci NOT NULL,
  `attachment_name` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `attachment_type` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `created` datetime NOT NULL,
  `last_change` datetime NOT NULL,
  `deadline` datetime NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `request_attachment` (
  `id` int(11) NOT NULL,
  `attachment_content` mediumblob NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `request_comments` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `request_id` int(11) NOT NULL,
  `username` varchar(20) COLLATE utf8_czech_ci NOT NULL,
  `science_comment` text COLLATE utf8_czech_ci NOT NULL,
  `science_score` int(11) NOT NULL,
  `computational_comment` text COLLATE utf8_czech_ci NOT NULL,
  `computational_score` int(11) NOT NULL,
  `socioeconomic_comment` text COLLATE utf8_czech_ci NOT NULL,
  `socioeconomic_score` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `request_id_username` (`request_id`,`username`)
) ENGINE=InnoDB AUTO_INCREMENT=43 DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `request_commissioner` (
  `request_id` int(11) NOT NULL,
  `username` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  UNIQUE KEY `unique_index` (`request_id`,`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

CREATE TABLE `setting` (
  `key` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `value` text COLLATE utf8_czech_ci NOT NULL,
  PRIMARY KEY (`key`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

7.2 Šablony

Portál obsahuje velké množství šablon a nemá smysl je v této práci uvádět a komentovat všechny. Pro příklad použití šablonovacího jazyka v nově vznikajícím informačním systému uvedeme jeden komplexní případ použití obsahující nejčastěji používané konstrukce na jednom místě:

```
<tbody tal:define="records
python:context.sql.project_requests_archive_list(pi_login=user.getUserName())">
<tr tal:condition="not:records" ><td i18n:translate="" colspan="9" align="center">You haven't any
project request.</td></tr>
<tr tal:repeat="record records">
<td tal:content="record/id" />
```

První ukázka je z pohledu zobrazujícího archiv projektových žádostí uživatele. Na prvním řádku je definována nová proměnná *records*, do které je následně vložen výsledek z SQL dotazu. Podmínka na následujícím řádku zajistí, že se při prázdném výsledku SQL dotazu zobrazí alespoň jeden řádek v tabulce, který uživateli oznámí, že v archivu nemá žádnou projektovou žádost. Pokud SQL metoda vrátí nějaká data, použijí se v konstrukci *tal:repeat*, která zajistí opakování řádku tabulky dle počtu záznamů z SQL metody. Poslední řádek pak ilustruje použití *tal:content*, kde je pro první buňku v opakujícím se řádku vložen identifikátor jako její obsah.

Další šablony jsou součástí přílohy.

V následující kapitole si popíšeme základ šablony.

7.2.1 Základ šablony

Základem šablony je následující blok:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
xmlns:i18n="http://xml.zope.org/namespaces/i18n"
lang="en"
metal:use-macro="here/main_template/macros/master"
i18n:domain="it4i.portal.common">
<head>
  <metal:slot fill-slot="base">
  <title tal:content="template/title" i18n:translate="">The title</title>
  <meta http-equiv="content-type" content="text/html;charset=utf-8">
</metal:slot>
</head>
<body>
```

Tato část i přes svou jednoduchou strukturu zajistí kompletní načtení všech potřebných javascriptů, stylopisů a dalších nezbytností, a to v přesně definovaném pořadí. Mimoto jsou zde také definovány

parametry pro překlad obsahu stránky pomocí *i18n* a vložen správný titulek nastavený jako parametr dané šablony.

Obsah následující v těle dokumentu je pak klasická HTML šablona využívající popsaný šablonovací systém.

7.3 Zdrojové kódy

Následující dvě podkapitoly věnujeme popisu nejdůležitějších částí aplikační vrstvy portálu, která sestává ze skriptů samotného portálu a dále z rozšiřujících balíčků nainstalovaných do portálu.

7.3.1 Skripty portálu

Stejně jako v předchozích příkladech je zbytečné vkládat všechny skripty do textu práce a tak uvedeme jeden případ jako ilustrační pro fungování skriptů v aplikačním serveru Zope.

Konkrétně se jedná o skript, který nabídne uživateli stažení přílohy projektové žádosti.

```
from it4i.portal.common.utils import get_user

request = container.REQUEST
response =  request.RESPONSE

username = get_user()

project_request_id = request.form["id"]
project_request_detail = context.sql.project_request_attachment(id=project_request_id,
                                                    pi_login=username
                                                   ).dictionaries()[0]

response.setHeader('content-type', project_request_detail['attachment_type'])
response.setHeader('Content-Disposition', 'attachment; filename="' +
project_request_detail['attachment_name'] + '"')
return project_request_detail['attachment_content'].decode('base64')
```

Na prvním řádku skriptu si nejdříve naimportujeme funkci *get_user* ze sdíleného balíčku, která nám pomocí Plone API zjistí přihlašovací jméno aktuálně přihlášeného uživatele. Následně získáme objekty reprezentující dotaz a odpověď uživateli, jeho uživatelské jméno a identifikátor odeslaný tomuto skriptu z HTML formuláře.

V dalším kroku spustíme SQL metodu pro získání obsahu projektové přílohy uložené v databázi. V této SQL metodě je pro lepší zabezpečení jako parametr uvedeno přihlašovací jméno uživatele.

Následně nastavíme návratové hodnoty *content-type* podle informací o příloze, které jsou rovněž uloženy v databázi, a pak vrátíme její obsah dekódován z formátu *base64*, který se používá při ukládání do databáze.

7.3.2 Rozšiřující balíčky

Rozšiřující balíčky, jak již bylo zmíněno výše, jsou jednou z možností, jak obejít omezení pro skripty realizované přímo z aplikačním serveru Zope. Tyto skripty jsou totiž zcela nezávislé a mohou tak používat všechny standardní knihovny (instalované společně s interpreterem jazyka Python) a také vytvářet knihovny vlastní.

Mimoto jsou tzv. vajíčka také používána jako zásuvné moduly pro aplikační server, což umožňuje nejen jejich sdílení v rámci několika navazujících projektů se společnou částí implementace, ale také jejich šíření napříč komunitou a vytváření tak jednotné sady nástrojů pro práci s externími službami a obsluhu běžných případů užití.

V našem případě jsme například jako jeden z volně šiřitelných modulů využili modul pro práci s adresářovou strukturou LDAP.

Mimoto byl během implementace vytvořen i náš vlastní modul sdružující společné části několika navazujících systémů - především společné JS knihovny, stylopisy, překlady proměnných apod. - ale také další doplňující funkce, které jsou buď společné pro několik systémů, nebo je nebylo možno přímo implementovat v rámci Zope.

Náš vlastní modul je stejně jako ostatní samostatně verzován a obsahuje v sobě všechny nezbytné informace. Je tedy jednoduché zjistit jeho současnou nainstalovanou verzi v portálu, provést sestavení nové verze a balíček pak na všech patřičných místech aktualizovat.

Jako příklad uvedeme funkci z našeho sdíleného modulu, která vytváří ZIP archiv při hromadném exportu příloh projektových žádostí.

```
def zip_attachments(attachments):
    import zipfile
    import cStringIO

    f = cStringIO.StringIO()
    with zipfile.ZipFile(f, mode='w',compression=zipfile.ZIP_DEFLATED) as zf:
        for attachment in attachments:
            zf.writestr(attachment['name'], attachment['content'])

    return f.getvalue()
```

Funkci jsou předány přílohy jako parametr a následně jsou s použitím standardních knihoven zabaleny do jednoho ZIP archivu s použitím speciálního objektu pro jeho uložení a obsah je pak navrácen zpět a nabídnut ke stažení jako odpověď na požadavek.

7.3.3 SQL Metody

Jako příklad pro prezentaci komplexnější SQL metody jsme vybrali tu, která zajišťuje editaci vztahu hodnotitele k určitému druhu výzvy. Nejdříve se danému uživateli odeberou práva k hodnocení všech výzev a následně se mu zpět nastaví právo k hodnocení výzev zadaných pomocí identifikátorů.

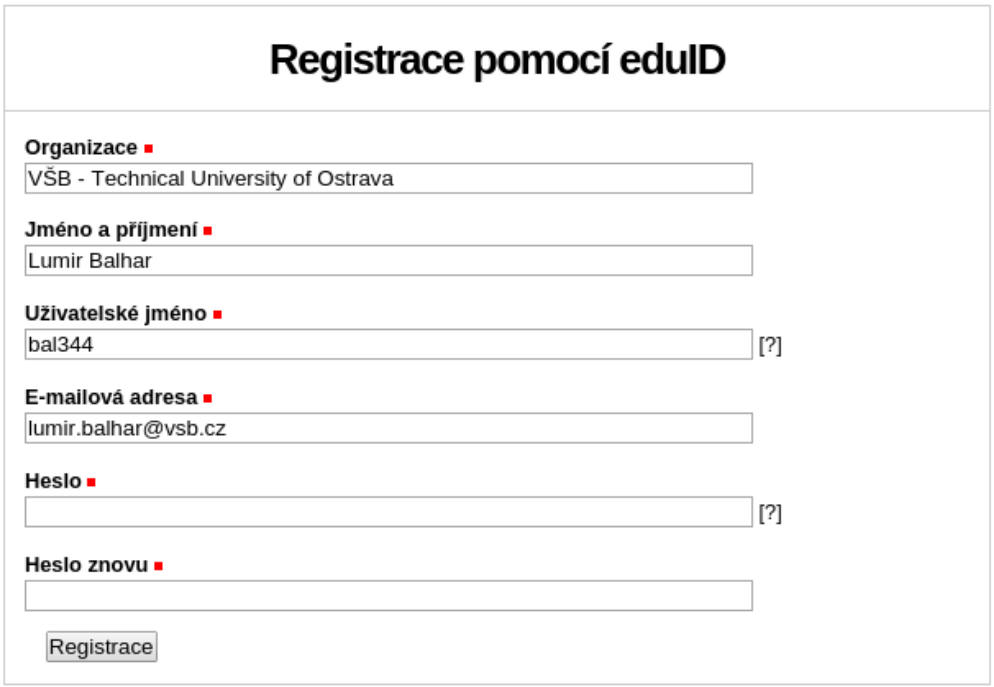
```
UPDATE evaluator_committee SET `<dtml-var alloc_comm_type>` = 0
<dtml-var sql_delimiter>
UPDATE evaluator_committee SET `<dtml-var alloc_comm_type>` = 1
WHERE <dtml-sqltest ids column=id type="string" multiple>
```

Na příkladu je možné vidět, že i díky použití DTML oddělovačů je možné použít více SQL dotazů v jedné metodě. Zároveň je zde vidět dvojí využití DTML. V prvním případě (*dtml-var*) jde o pouhé vložení proměnné na patřičné místo s ošetřením její podoby, zatímco v druhém případě (*dtml-sqltest*) je tato konstrukce použita k ověření podmínky v části WHERE, kde je předána nejen proměnná, ale také název odpovídajícího sloupce v tabulce a parametr *multiple*, který zajistí zpracování i více položek v jedné proměnné.

8. Screenshoty aplikace

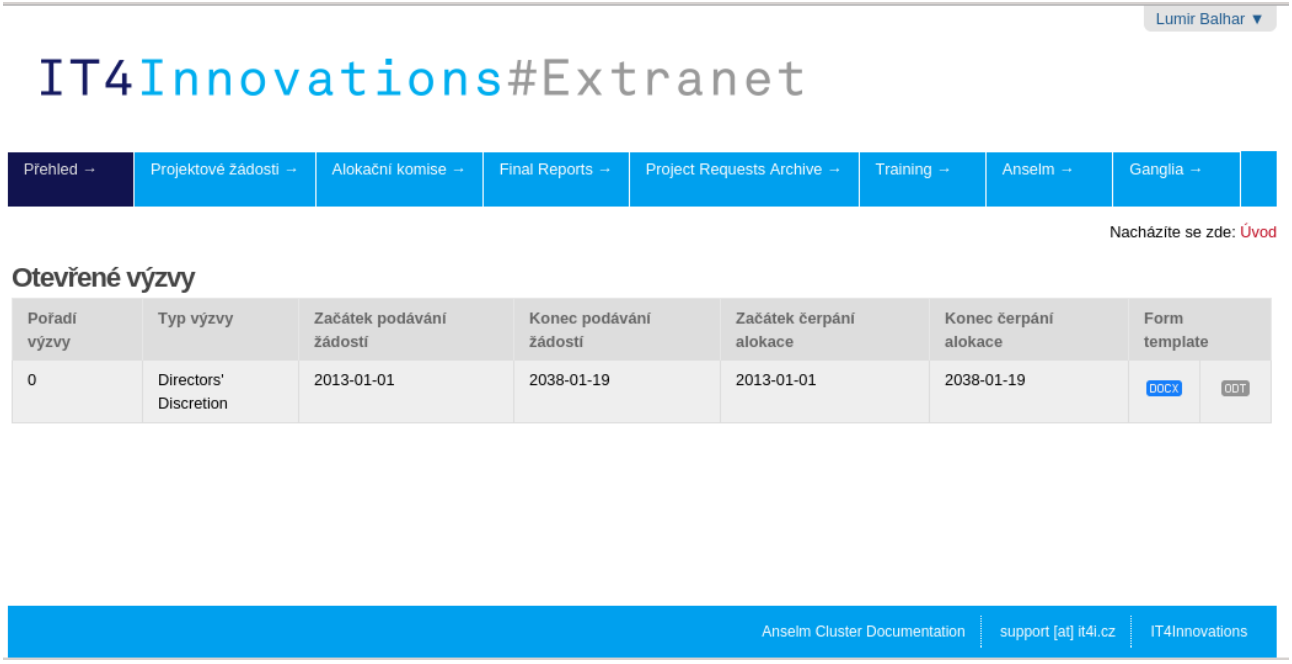
Obrázek 4 Volba způsobu registrace

Obrázek 5 Registrační formulář

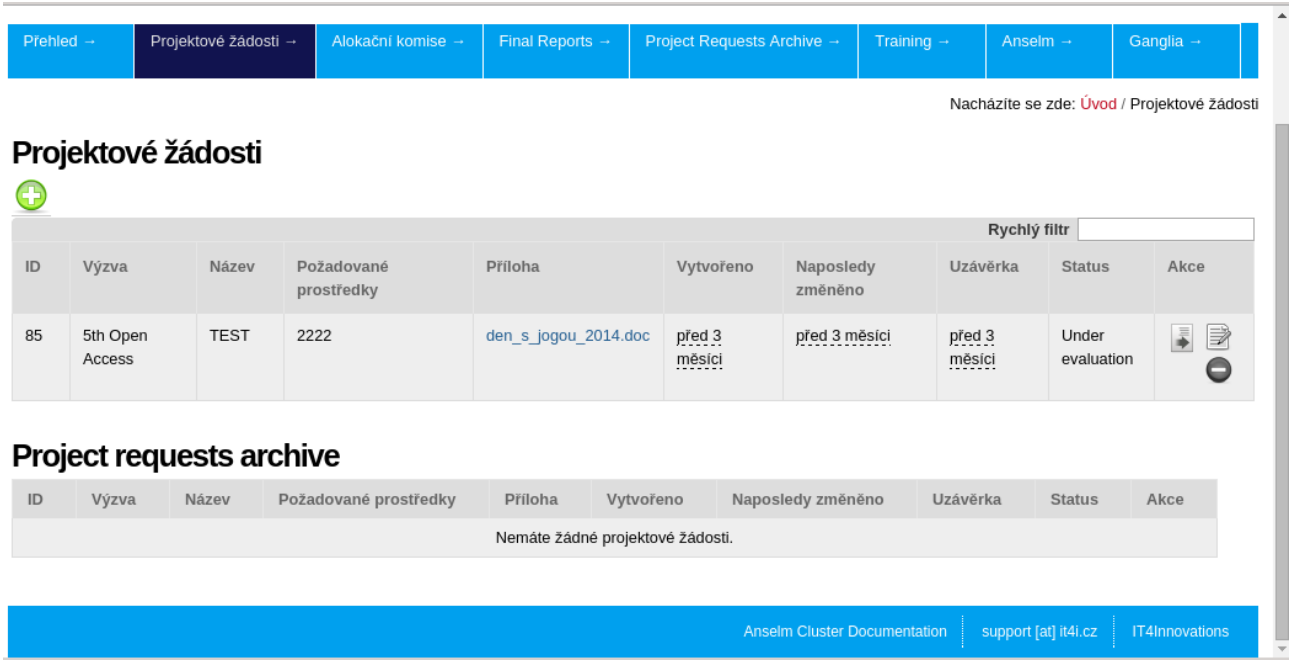


Obrázek 6 Přihlašovací formulář

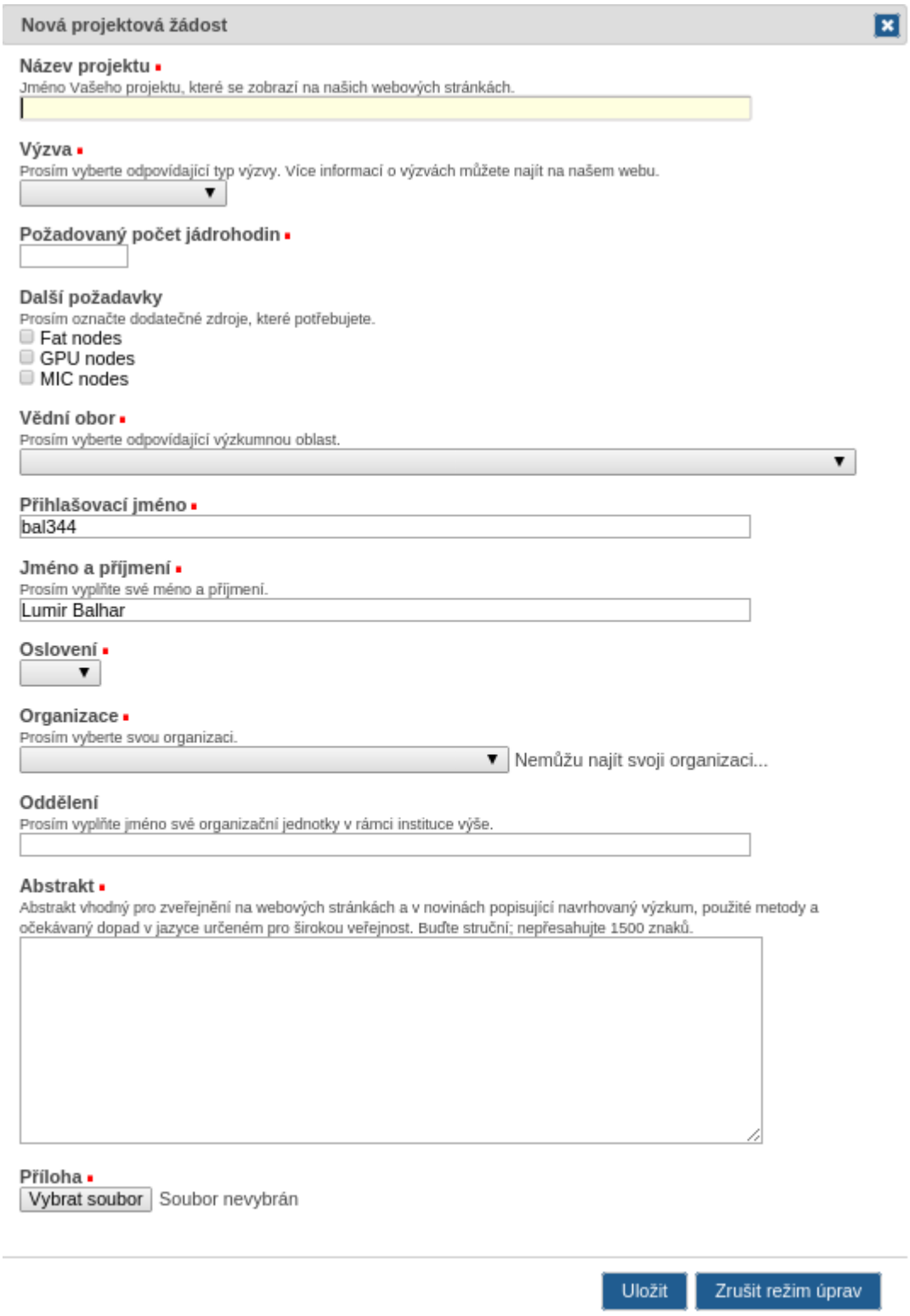
Obrázek 7 Přehledová stránka s otevřenými výzvami



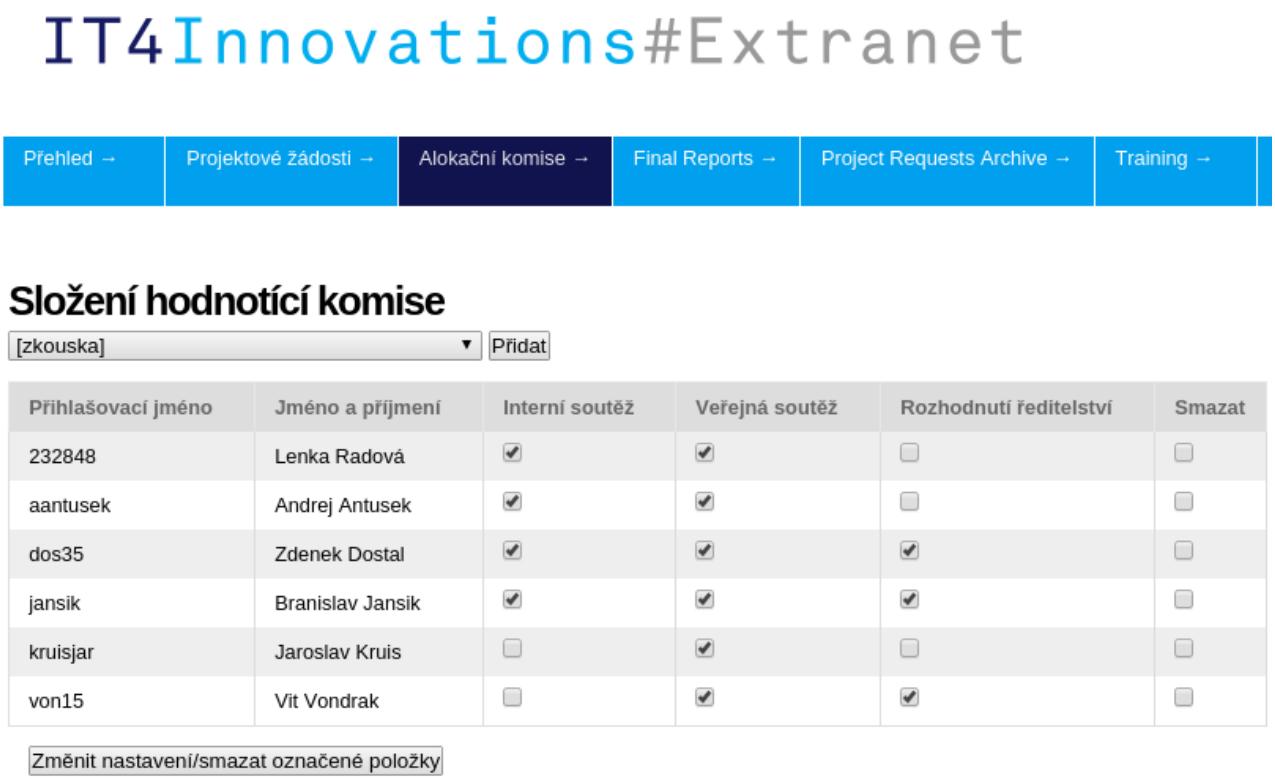
Obrázek 8 Projektové žádosti uživatele



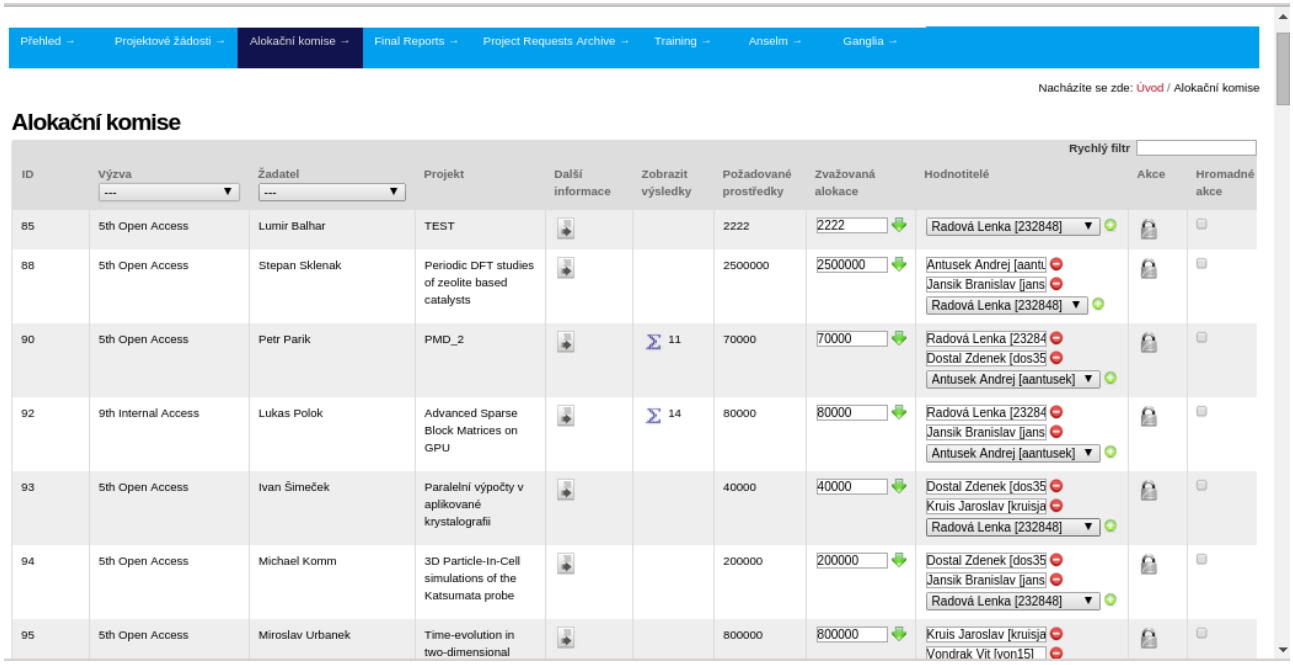
Obrázek 9 Formulář pro novou žádost



Obrázek 10 Tvorba hodnotící komise



Obrázek 11 Rozhraní alokační komise



9. Závěr

Při zpětném pohledu mohu označit vývoj této aplikace za úspěšný a zvolené metody a komponenty za správné a vyhovující. Aplikace je v ostrém provozu používána jak vedením superpočítačového centra, tak i běžnými uživateli z ČR i zahraničí a svým uživatelům zlepšuje komfort práce a usnadňuje provádění rutinních úkolů.

Zvolené nástroje a podpůrné systémy umožňují v případě zvýšeného počtu uživatelů snadno a rychle škálovat informační systém na všech úrovních.

Také vývoj informačního systému je díky zvoleným nástrojům a metodám možno snadno přenést na další členy a rozšířit tím tým vývojářů.

Zvolená metodika vývoje pomocí FDD je pro tento typ aplikace vhodná, protože pravděpodobně nikdy nedojde do stavu, kdy bude označena za hotovou, ale neustále se budou od uživatelů objevovat požadavky na rozšíření či vylepšení funkcionality.

S ohledem na výše zmíněné je i tato diplomová práce popisem informačního systému v určité fázi jeho vývoje, který se během její přípravy měnil v závislosti na požadavcích uživatelů a vedení.

10. Literatura

[1] eduID - Česká akademická federace identit eduID.cz [1. 5. 2015] https://www.eduid.cz/

[2] Shibboleth - Shibboleth [1. 5. 2015] https://shibboleth.net/

[3] Zope - Zope [1. 5. 2015] http://zope.org/

[4] Plone - Plone CMS: Open Source Content Management [1. 5. 2015] https://plone.org/

[5] Gitlab - Create, review and deploy code together [1. 5. 2015] https://about.gitlab.com/

[6] Apache - The Apache HTTP Server Project [1. 5. 2015] http://httpd.apache.org/

[7] MySQL - The world's most popular open source database [1. 5. 2015] https://www.mysql.com/

11. Přílohy

Přílohou této diplomové práce je CD obsahující následující součásti:

a) Práci samotnou ve formátech DOCX a PDF.

b) Zdrojové kódy popisované aplikace ve složce source codes.